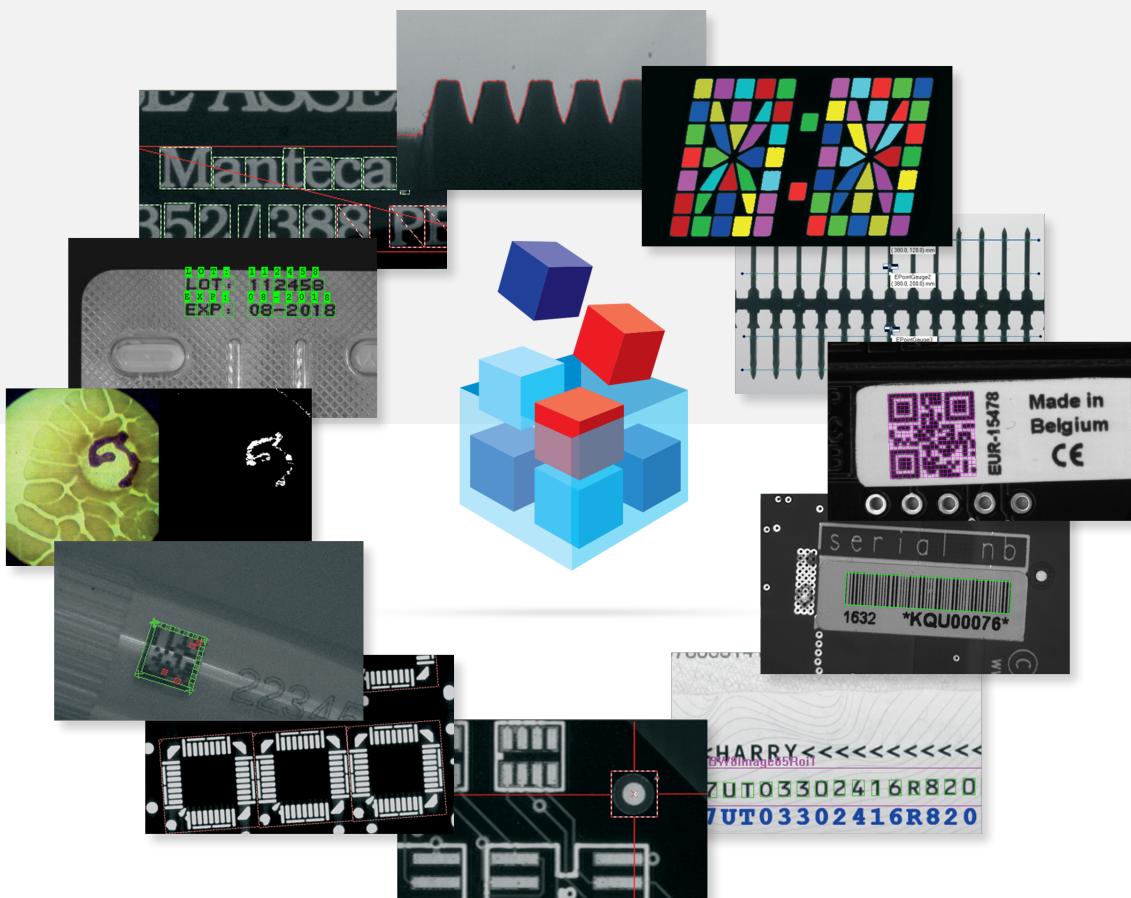


REFERENCE

Open eVision

ActiveX 2.1.0



Terms of Use

EURESYS s.a. shall retain all property rights, title and interest of the documentation of the hardware and the software, and of the trademarks of EURESYS s.a.

All the names of companies and products mentioned in the documentation may be the trademarks of their respective owners.

The licensing, use, leasing, loaning, translation, reproduction, copying or modification of the hardware or the software, brands or documentation of EURESYS s.a. contained in this book, is not allowed without prior notice.

EURESYS s.a. may modify the product specification or change the information given in this documentation at any time, at its discretion, and without prior notice.

EURESYS s.a. shall not be liable for any loss of or damage to revenues, profits, goodwill, data, information systems or other special, incidental, indirect, consequential or punitive damages of any kind arising in connection with the use of the hardware or the software of EURESYS s.a. or resulting of omissions or errors in this documentation.

This documentation is provided with Open eVision 2.1.0 (doc build 2017-06-21).
© 2017 EURESYS s.a.

Contents

Libraries	64
Easy3D Library	64
EasyImage Library	65
EasyColor Library	65
EasyObject Library	66
EasyMatch Library	67
EasyFind Library	67
EasyGauge Library	68
EasyOCR Library	68
EasyOCV Library	69
EasyBarCode Library	69
EasyMatrixCode Library	70
EasyQRCode Library	70
Legacy	71
EasyObject Library (Legacy)	71
Classes	72
E3DAffineTransformer Class	72
E3DAffineTransformer.AddAnisotropicScalingTransform	75
E3DAffineTransformer.AddIsotropicScalingTransform	76
E3DAffineTransformer.AddOrthographicProjectionTransform	76
E3DAffineTransformer.AddPerspectiveProjectionTransform	77
E3DAffineTransformer.AddRotationXTransform	77
E3DAffineTransformer.AddRotationYTransform	78
E3DAffineTransformer.AddRotationZTransform	78
E3DAffineTransformer.AddTransform	79
E3DAffineTransformer.AddTranslationTransform	79
E3DAffineTransformer.ApplyMatrix	80
E3DAffineTransformer.ApplyTransform	80
E3DAffineTransformer.CreateAnisotropicScalingMatrix	81
E3DAffineTransformer.CreateIdentityMatrix	81
E3DAffineTransformer.CreateIsotropicScalingMatrix	82
E3DAffineTransformer.CreateOrthographicProjectionMatrix	82
E3DAffineTransformer.CreatePerspectiveProjectionMatrix	83
E3DAffineTransformer.CreateRotationXMatrix	83
E3DAffineTransformer.CreateRotationYMatrix	84
E3DAffineTransformer.CreateRotationZMatrix	84

E3DAffineTransformer.CreateTranslationMatrix	85
E3DAffineTransformer.E3DAffineTransformer	85
E3DAffineTransformer.Reset	85
E3DAffineTransformer.Transform	86
E3DAffineTransformMatrix Class	86
E3DAffineTransformMatrix.E3DAffineTransformMatrix	87
E3DAffineTransformMatrix.GetValue	87
E3DAffineTransformMatrix.operator*	87
E3DAffineTransformMatrix.SetValue	88
E3DDepthMapToPointCloud Class	89
E3DDepthMapToPointCloud.Apply	89
E3DDepthMapToPointCloud.E3DDepthMapToPointCloud	90
E3DExplicitGeometricCalibration Class	91
E3DExplicitGeometricCalibration.Apply	92
E3DExplicitGeometricCalibration.CameraAngle	93
E3DExplicitGeometricCalibration.CameraPlaneDistance	93
E3DExplicitGeometricCalibration.E3DExplicitGeometricCalibration	94
E3DExplicitGeometricCalibration.FocalLength	95
E3DExplicitGeometricCalibration.MotionIncrement	95
E3DExplicitGeometricCalibration.PlaneAngle	95
E3DExplicitGeometricCalibration.SensorHeight	96
E3DExplicitGeometricCalibration.SensorWidth	96
E3DPointCloud Class	96
E3DPointCloud.AddPoint	98
E3DPointCloud.AddPointCloud	98
E3DPointCloud.AddPoints	99
E3DPointCloud.Clear	99
E3DPointCloud.E3DPointCloud	99
E3DPointCloud.FillPointsBuffer	100
E3DPointCloud.GetPoint	100
E3DPointCloud.LoadPCD	101
E3DPointCloud.NumPoints	101
E3DPointCloud.Points	101
E3DPointCloud.PointsBuffer	102
E3DPointCloud.SavePCD	102
E3DPointCloud.Serialize	102
E3DPointCloudFactory Class	103
E3DPointCloudFactory.CreateCubicPointCloud	103
E3DPointCloudFactory.CreateRectangularPointCloud	104
E3DPointCloudFactory.CreateSphericPointCloud	105
E3DPointCloudStatistics Class	106
E3DPointCloudStatistics.GetPointCloudBounds	106
E3DPointCloudStatistics.GetPointCloudCenterOfGravity	107
E3DPointCloudViewer Class	108
E3DPointCloudViewer.ConfigureViewport	108
E3DPointCloudViewer.E3DPointCloudViewer	109
E3DPointCloudViewer.SetCloud	110

E3DPointCloudViewer.SetPosition	110
E3DPointCloudViewer.SetViewingAngle	111
E3DPointCloudViewer.Show	112
E3DRectangularCropper Class	112
E3DRectangularCropper.Crop	112
E3DRectangularCropper.E3DRectangularCropper	113
E3DSimpleCropper Class	114
E3DSimpleCropper.Crop	115
E3DSimpleCropper.E3DSimpleCropper	115
E3DSimpleCropper.XRange	116
E3DSimpleCropper.YRange	116
E3DSimpleCropper.ZRange	116
E3DSphericalCropper Class	117
E3DSphericalCropper.Crop	117
E3DSphericalCropper.E3DSphericalCropper	118
Easy Class	118
Easy.AngleUnit	121
Easy.CheckLicense	121
Easy.CheckLicenses	121
Easy.CheckOemKey	122
Easy.CloseImageGraphicContext	122
Easy.FromRadians	123
Easy.GetBestMatchingImageType	123
Easy.GetErrorText	124
Easy.OpenImageGraphicContext	124
Easy.Render3D	125
Easy.RenderColorHistogram	127
Easy.Resize	128
Easy.SetOemKey	129
Easy.StartTiming	130
Easy.StopTiming	130
Easy.ToRadians	130
Easy.TrueTimingResolution	131
Easy.Version	131
EasyColor Class	132
EasyColor.AssignNearestClass	139
EasyColor.AssignNearestClassCenter	140
EasyColor.BayerToC24	140
EasyColor.C24ToBayer	141
EasyColor.CieAB	142
EasyColor.CieAG	142
EasyColor.CieAR	143
EasyColor.CieD50B	143
EasyColor.CieD50G	143
EasyColor.CieD50R	144
EasyColor.CieD55B	144
EasyColor.CieD55G	144
EasyColor.CieD55R	145

EasyColor.CieD65B	145
EasyColor.CieD65G	145
EasyColor.CieD65R	146
EasyColor.CieFB	146
EasyColor.CieFG	146
EasyColor.CieFR	147
EasyColor.ClassAverages	147
EasyColor.ClassVariances	148
EasyColor.CompensateNtscGamma	149
EasyColor.CompensatePalGamma	149
EasyColor.CompensateSmpteGamma	149
EasyColor.Compose	150
EasyColor.Decompose	151
EasyColor.Dequantize	152
EasyColor.DstQuantization	153
EasyColor.Format422To444	154
EasyColor.Format444To422	155
EasyColor.GetComponent	155
EasyColor.ImproveClassCenters	156
EasyColor.IshToRgb	157
EasyColor.LabToRgb	158
EasyColor.LabToXyz	158
EasyColor.LchToRgb	159
EasyColor.LshToRgb	160
EasyColor.LuvToRgb	160
EasyColor.LuvToXyz	161
EasyColor.NtscGamma	162
EasyColor.PalGamma	162
EasyColor.PseudoColor	162
EasyColor.Quantize	163
EasyColor.RegisterPlanes	165
EasyColor.RgbStandard	166
EasyColor.RgbTolsh	166
EasyColor.RgbToLab	167
EasyColor.RgbToLch	168
EasyColor.RgbTolsh	168
EasyColor.RgbToLuv	169
EasyColor.RgbToReducedXyz	170
EasyColor.RgbToVsh	170
EasyColor.RgbToXyz	171
EasyColor.RgbToYiq	172
EasyColor.RgbToYsh	172
EasyColor.RgbToYuv	173
EasyColor.SetComponent	174
EasyColor.SmpteGamma	175
EasyColor.SrcQuantization	175
EasyColor.Transform	175
EasyColor.TransformBayer	176
EasyColor.VshToRgb	177

EasyColor.XyzToLab	178
EasyColor.XyzToLuv	178
EasyColor.XyzToRgb	179
EasyColor.YiqToRgb	180
EasyColor.YshToRgb	180
EasyColor.YuvToRgb	181
EasyImage Class	182
EasylImage.AdaptiveThreshold	200
EasylImage.AnalyseHistogram	201
EasylImage.AnalyseHistogramBW16	201
EasylImage.Area	202
EasylImage.AreaDoubleThreshold	203
EasylImage.ArgumentImage	204
EasylImage.AutoThreshold	205
EasylImage.BiLevelBlackTopHatBox	207
EasylImage.BiLevelBlackTopHatDisk	208
EasylImage.BiLevelCloseBox	208
EasylImage.BiLevelCloseDisk	209
EasylImage.BiLevelDilateBox	210
EasylImage.BiLevelDilateDisk	210
EasylImage.BiLevelErodeBox	211
EasylImage.BiLevelErodeDisk	212
EasylImage.BiLevelMedian	212
EasylImage.BiLevelMorphoGradientBox	213
EasylImage.BiLevelMorphoGradientDisk	214
EasylImage.BiLevelOpenBox	214
EasylImage.BiLevelOpenDisk	215
EasylImage.BiLevelThick	216
EasylImage.BiLevelThin	217
EasylImage.BiLevelWhiteTopHatBox	218
EasylImage.BiLevelWhiteTopHatDisk	219
EasylImage.BinaryMoments	219
EasylImage.BlackTopHatBox	222
EasylImage.BlackTopHatDisk	223
EasylImage.CloseBox	224
EasylImage.CloseDisk	225
EasylImage.Contour	226
EasylImage.Convert	228
EasylImage.ConvertTo422	233
EasylImage.ConvolGaussian	234
EasylImage.ConvolGradient	235
EasylImage.ConvolGradientX	236
EasylImage.ConvolGradientY	237
EasylImage.ConvolHighpass1	238
EasylImage.ConvolHighpass2	239
EasylImage.ConvolKernel	239
EasylImage.ConvolLaplacian4	240
EasylImage.ConvolLaplacian8	241
EasylImage.ConvolLaplacianX	242

EasylImage.ConvolveLaplacianY	243
EasylImage.ConvolveLowpass1	244
EasylImage.ConvolveLowpass2	244
EasylImage.ConvolveLowpass3	245
EasylImage.ConvolvePrewitt	246
EasylImage.ConvolvePrewittX	247
EasylImage.ConvolvePrewittY	248
EasylImage.ConvolveRoberts	249
EasylImage.ConvolveSobel	249
EasylImage.ConvolveSobelX	250
EasylImage.ConvolveSobelY	251
EasylImage.ConvolveSymmetricKernel	252
EasylImage.ConvolveUniform	253
EasylImage.Copy	254
EasylImage.CumulateHistogram	256
EasylImage.DilateBox	257
EasylImage.DilateDisk	258
EasylImage.Distance	259
EasylImage.DoubleThreshold	259
EasylImage.Equalize	261
EasylImage.ErodeBox	261
EasylImage.ErodeDisk	263
EasylImage.Focusing	264
EasylImage.Gain	264
EasylImage.GainOffset	265
EasylImage.GetFrame	266
EasylImage.GetProfilePeaks	267
EasylImage.GradientScalar	269
EasylImage.GravityCenter	270
EasylImage.HDRFusion	271
EasylImage.Histogram	272
EasylImage.HistogramThreshold	274
EasylImage.HistogramThresholdBW16	275
EasylImage.HitAndMiss	276
EasylImage.HorizontalMirror	277
EasylImage.ImageToLineSegment	277
EasylImage.ImageToPath	279
EasylImage.IsodataThreshold	281
EasylImage.IsodataThresholdBW16	281
EasylImage.LinearTransform	282
EasylImage.LineSegmentToImage	284
EasylImage.LocalAverage	286
EasylImage.LocalDeviation	287
EasylImage.Lut	288
EasylImage.MatchFrames	289
EasylImage.Median	290
EasylImage.ModulusImage	291
EasylImage.MorphoGradientBox	292
EasylImage.MorphoGradientDisk	293

EasylImage.Normalize	294
EasylImage.Offset	295
EasylImage.OpenBox	295
EasylImage.OpenDisk	297
EasylImage.Oper	298
EasylImage.Overlay	302
EasylImage.OverlayColor	304
EasylImage.PathToImage	304
EasylImage.PixelAverage	305
EasylImage.PixelCompare	307
EasylImage.PixelCount	308
EasylImage.PixelMax	310
EasylImage.PixelMaxBW16	311
EasylImage.PixelMaxBW8	312
EasylImage.PixelMin	312
EasylImage.PixelMinBW16	313
EasylImage.PixelMinBW8	314
EasylImage.PixelStat	314
EasylImage.PixelStatBW16	315
EasylImage.PixelStatBW8	316
EasylImage.PixelStdDev	317
EasylImage.PixelVariance	319
EasylImage.ProfileDerivative	321
EasylImage.ProjectOnAColumn	322
EasylImage.ProjectOnARow	324
EasylImage.RealignFrame	325
EasylImage.RebuildFrame	327
EasylImage.RecursiveAverage	328
EasylImage.Register	329
EasylImage.RmsNoise	334
EasylImage.ScaleRotate	336
EasylImage.SetCircleWarp	338
EasylImage.SetFrame	339
EasylImage.SetRecursiveAverageLUT	340
EasylImage.SetupEqualize	341
EasylImage.Shrink	341
EasylImage.SignalNoiseRatio	342
EasylImage.SwapFrames	344
EasylImage.Thick	345
EasylImage.Thin	346
EasylImage.ThreeLevelsMinResidueThreshold	347
EasylImage.Threshold	348
EasylImage.TwoLevelsMinResidueThreshold	351
EasylImage.Uniformize	352
EasylImage.VerticalMirror	356
EasylImage.Warp	356
EasylImage.WeightedMoments	357
EasylImage.WhiteTopHatBox	364
EasylImage.WhiteTopHatDisk	365

EasyObject Class	366
EasyObject.ContourArea	367
EasyObject.ContourGravityCenter	367
EasyObject.ContourInertia	368
EasyObject.IsFloatFeature	369
EasyObject.IsIntegerFeature	369
EasyObject.IsUnsignedIntegerFeature	370
EBarcode Class	370
EBarcode.AdditionalSymbologies	374
EBarcode.Decode	374
EBarcode.Detect	375
EBarcode.Drag	375
EBarcode.Draw	376
EBarcode.DrawWithCurrentPen	377
EBarcode.EBarcode	378
EBarcode.GetDecodedAngle	378
EBarcode.GetDecodedDirection	379
EBarcode.GetDecodedRectangle	380
EBarcode.GetDecodedSymbology	380
EBarcode.GetSymbologyName	381
EBarcode.HitTest	381
EBarcode.KnownLocation	382
EBarcode.KnownModule	382
EBarcode.Module	383
EBarcode.NumDecodedSymbologies	383
EBarcode.NumEnabledSymbologies	384
EBarcode.Read	384
EBarcode.Rectangle	385
EBarcode.RelativeReadingSizeX	385
EBarcode.RelativeReadingSizeY	385
EBarcode.RelativeReadingX	386
EBarcode.RelativeReadingY	386
EBarcode.SetReadingCenter	386
EBarcode.SetReadingSize	387
EBarcode.StandardSymbologies	387
EBarcode.ThicknessRatio	388
EBarcode.VerifyChecksum	388
EBaseROI Class	389
EBaseROI.Attach	395
EBaseROI.Author	395
EBaseROI.BaseTopParent	396
EBaseROI.BitsPerPixel	396
EBaseROI.ColorSystem	396
EBaseROI.ColPitch	397
EBaseROI.Comment	397
EBaseROI.CopyTo	398
EBaseROI.CropToImage	398
EBaseROI.Date	399

EBaseROI.Drag399
EBaseROI.Draw400
EBaseROI.DrawFrame402
EBaseROI.DrawFrameWithCurrentPen404
EBaseROI.FirstSubROI405
EBaseROI.GetImagePtr405
EBaseROI.GetSubBaseROIs406
EBaseROI.HasSubROI407
EBaseROI.Height407
EBaseROI.HitTest407
EBaseROI.IsAnROI408
EBaseROI.IsVoid409
EBaseROI.Load409
EBaseROI.NextSiblingROI410
EBaseROI.OrgX410
EBaseROI.OrgY411
EBaseROI.Parent411
EBaseROI.PlanesPerPixel411
EBaseROI.RowPitch412
EBaseROI.Save412
EBaseROI.SaveJpeg413
EBaseROI.SaveJpeg2K413
EBaseROI.Serialize414
EBaseROI.SetImagePtr414
EBaseROI.SetPlacement415
EBaseROI.SetSize416
EBaseROI.Title417
EBaseROI.TotalHeight417
EBaseROI.TotalOrgX418
EBaseROI.TotalOrgY418
EBaseROI.TotalWidth419
EBaseROI.Type419
EBaseROI.Width419
EBinaryImageSegmenter Class420
EBW16PathVector Class420
EBW16PathVector.AddElement422
EBW16PathVector.Closed422
EBW16PathVector.Draw422
EBW16PathVector.DrawWithCurrentPen424
EBW16PathVector.EBW16PathVector425
EBW16PathVector.GetElement425
EBW16PathVector.operator[]426
EBW16PathVector.operator=426
EBW16PathVector.RawDataPtr427
EBW16PathVector.SetElement427
EBW16Vector Class428
EBW16Vector.AddElement429
EBW16Vector.Draw429

EBW16Vector.DrawWithCurrentPen	431
EBW16Vector.EBW16Vector	432
EBW16Vector.GetElement	432
EBW16Vector.operator[]	433
EBW16Vector.operator=	433
EBW16Vector.RawDataPtr	434
EBW16Vector.SetElement	434
EBW16Vector.WeightedMoment	434
EBW32Vector Class	435
EBW32Vector.AddElement	437
EBW32Vector.Draw	437
EBW32Vector.DrawWithCurrentPen	438
EBW32Vector.EBW32Vector	439
EBW32Vector.GetElement	440
EBW32Vector.operator[]	440
EBW32Vector.operator=	441
EBW32Vector.RawDataPtr	441
EBW32Vector.SetElement	442
EBW32Vector.WeightedMoment	442
EBW8PathVector Class	443
EBW8PathVector.AddElement	444
EBW8PathVector.Closed	445
EBW8PathVector.Draw	445
EBW8PathVector.DrawWithCurrentPen	446
EBW8PathVector.EBW8PathVector	447
EBW8PathVector.GetElement	447
EBW8PathVector.operator[]	448
EBW8PathVector.operator=	448
EBW8PathVector.RawDataPtr	449
EBW8PathVector.SetElement	449
EBW8Vector Class	450
EBW8Vector.AddElement	451
EBW8Vector.Draw	452
EBW8Vector.DrawWithCurrentPen	453
EBW8Vector.EBW8Vector	454
EBW8Vector.GetElement	454
EBW8Vector.operator[]	455
EBW8Vector.operator=	455
EBW8Vector.RawDataPtr	456
EBW8Vector.SetElement	456
EBW8Vector.WeightedMoment	456
EBWHistogramVector Class	457
EBWHistogramVector.AddElement	458
EBWHistogramVector.Draw	459
EBWHistogramVector.DrawWithCurrentPen	460
EBWHistogramVector.EBWHistogramVector	461
EBWHistogramVector.GetElement	461
EBWHistogramVector.operator[]	462

EBWHistogramVector.operator=	462
EBWHistogramVector.RawDataPtr	463
EBWHistogramVector.SetElement	463
EC24PathVector Class	464
EC24PathVector.AddElement	465
EC24PathVector.Closed	466
EC24PathVector.Draw	466
EC24PathVector.DrawLineWithCurrentPen	467
EC24PathVector.EC24PathVector	468
EC24PathVector.GetElement	468
EC24PathVector.operator[]	469
EC24PathVector.operator=	469
EC24PathVector.RawDataPtr	470
EC24PathVector.SetElement	470
EC24Vector Class	471
EC24Vector.AddElement	472
EC24Vector.Draw	473
EC24Vector.EC24Vector	475
EC24Vector.GetElement	476
EC24Vector.operator[]	476
EC24Vector.operator=	477
EC24Vector.RawDataPtr	477
EC24Vector.SetElement	477
ECannyEdgeDetector Class	478
ECannyEdgeDetector.Apply	479
ECannyEdgeDetector.ECannyEdgeDetector	480
ECannyEdgeDetector.HighThreshold	480
ECannyEdgeDetector.LowThreshold	481
ECannyEdgeDetector.ResetSmoothingScale	481
ECannyEdgeDetector.SmoothingScale	481
ECannyEdgeDetector.ThresholdingMode	482
EChecker Class	482
EChecker.AddPathName	486
EChecker.Attach	487
EChecker.Average	487
EChecker.BatchLearn	488
EChecker.DarkGray	488
EChecker.DegreesOfFreedom	489
EChecker.Deviation	489
EChecker.Drag	489
EChecker.Draw	490
EChecker.DrawLineWithCurrentPen	491
EChecker.EChecker	492
EChecker.EmptyPathNames	493
EChecker.High	493
EChecker.HitHandle	494
EChecker.HitRoi	494
EChecker.HitTest	494

EChecker.Learn	495
EChecker.LightGray	496
EChecker.Load	496
EChecker.Low	497
EChecker.Normalize	497
EChecker.NumAverageSamples	498
EChecker.NumDeviationSamples	498
EChecker.PanX	498
EChecker.PanY	499
EChecker.Register	499
EChecker.Registered	499
EChecker.RelativeTolerance	500
EChecker.Save	500
EChecker.SetPan	501
EChecker.SetTolerance	501
EChecker.SetZoom	502
EChecker.ToleranceX	502
EChecker.ToleranceY	503
EChecker.ZoomX	503
EChecker.ZoomY	503
ECircle Class	504
ECircle.Amplitude	507
ECircle.Apex	507
ECircle.ApexAngle	507
ECircle.ArcLength	508
ECircle.CopyTo	508
ECircle.Diameter	509
ECircle.Direct	509
ECircle.DirectInternal	510
ECircle.Distance	510
ECircle.ECircle	511
ECircle.End	512
ECircle.EndAngle	512
ECircle.Full	513
ECircle.GetDistanceBetweenLineAndCircle	513
ECircle.GetDistanceBetweenPointAndCircle	514
ECircle.GetIntersectionOfCircles	515
ECircle.GetIntersectionOfLineAndCircle	515
ECircle.GetPoint	516
ECircle.GetProjectionOfPointOnCircle	517
ECircle.operator=	517
ECircle.Org	518
ECircle.OrgAngle	518
ECircle.Radius	518
ECircle.Serialize	519
ECircle.SetFromCenterAndOrigin	519
ECircle.SetFromOriginMiddleEnd	520
ECircleGauge Class	521

ECircleGauge.Active	526
ECircleGauge.AddSkipRange	527
ECircleGauge.AverageDistance	527
ECircleGauge.Circle	528
ECircleGauge.CopyTo	528
ECircleGauge.DisableInnerFiltering	529
ECircleGauge.Drag	529
ECircleGauge.Draw	530
ECircleGauge.DrawWithCurrentPen	531
ECircleGauge.ECircleGauge	531
ECircleGauge.FilteringThreshold	532
ECircleGauge.GetMeasuredPeak	532
ECircleGauge.GetMeasuredPoint	533
ECircleGauge.GetMinNumFitSamples	534
ECircleGauge.GetSample	534
ECircleGauge.GetSkipRange	535
ECircleGauge.HitTest	536
ECircleGauge.HVConstraint	536
ECircleGauge.InnerFilteringEnabled	536
ECircleGauge.InnerFilteringThreshold	537
ECircleGauge.Measure	537
ECircleGauge.MedasuredCircle	538
ECircleGauge.MeasureSample	538
ECircleGauge.MeasureWithoutFitting	539
ECircleGauge.MinAmplitude	539
ECircleGauge.MinArea	540
ECircleGauge.NumFilteringPasses	540
ECircleGauge.NumMeasuredPoints	540
ECircleGauge.NumSamples	541
ECircleGauge.NumSkipRanges	541
ECircleGauge.NumValidSamples	541
ECircleGauge.operator=	542
ECircleGauge.Plot	542
ECircleGauge.PlotWithCurrentPen	544
ECircleGauge.Process	545
ECircleGauge.RectangularSamplingArea	545
ECircleGauge.RemoveAllSkipRanges	546
ECircleGauge.RemoveSkipRange	546
ECircleGauge.SamplingStep	546
ECircleGauge.SetMinNumFitSamples	547
ECircleGauge.Shape	548
ECircleGauge.Smoothing	548
ECircleGauge.Thickness	548
ECircleGauge.Threshold	549
ECircleGauge.Tolerance	549
ECircleGauge.TransitionChoice	549
ECircleGauge.TransitionIndex	550
ECircleGauge.TransitionType	550
ECircleGauge.Type	550

ECircleGauge.Valid551
ECircleShape Class551
ECircleShape.Amplitude554
ECircleShape.Angle554
ECircleShape.Apex555
ECircleShape.ApexAngle555
ECircleShape.Arclength555
ECircleShape.Center556
ECircleShape.CenterX556
ECircleShape.CenterY556
ECircleShape.Circle557
ECircleShape.Closest557
ECircleShape.CopyTo557
ECircleShape.Diameter558
ECircleShape.Direct558
ECircleShape.Drag559
ECircleShape.Draw559
ECircleShape.DrawWithCurrentPen560
ECircleShape.End561
ECircleShape.EndAngle561
ECircleShape.Full561
ECircleShape.GetPoint562
ECircleShape.HitTest562
ECircleShape.operator=562
ECircleShape.Org563
ECircleShape.OrgAngle563
ECircleShape.Radius563
ECircleShape.Scale564
ECircleShape.SetCenterXY564
ECircleShape.SetFromCenterAndOrigin565
ECircleShape.SetFromOriginMiddleEnd565
ECircleShape.Type566
ECodedElement Class566
ECodedElement.Area575
ECodedElement.AsHole575
ECodedElement.AsObject575
ECodedElement.BottomLimit576
ECodedElement.BoundingBox576
ECodedElement.BoundingBoxCenter577
ECodedElement.BoundingBoxCenterX577
ECodedElement.BoundingBoxCenterY577
ECodedElement.BoundingBoxHeight578
ECodedElement.BoundingBoxWidth578
ECodedElement.ComputeConvexHull578
ECodedElement.ComputeFeretBox579
ECodedElement.ComputePixelGrayAverage579
ECodedElement.ComputePixelGrayDeviation580
ECodedElement.ComputePixelGrayVariance580

ECodedElement.ComputePixelMax580
ECodedElement.ComputePixelMin581
ECodedElement.ComputeWeightedGravityCenter581
ECodedElement.Contour582
ECodedElement.ContourX582
ECodedElement.ContourY582
ECodedElement.Eccentricity583
ECodedElement.ElementIndex583
ECodedElement.EllipseAngle584
ECodedElement.EllipseHeight584
ECodedElement.EllipseWidth584
ECodedElement.FeretBox22Box585
ECodedElement.FeretBox22Center585
ECodedElement.FeretBox22CenterX585
ECodedElement.FeretBox22CenterY586
ECodedElement.FeretBox22Height586
ECodedElement.FeretBox22Width586
ECodedElement.FeretBox45Box587
ECodedElement.FeretBox45Center587
ECodedElement.FeretBox45CenterX587
ECodedElement.FeretBox45CenterY588
ECodedElement.FeretBox45Height588
ECodedElement.FeretBox45Width588
ECodedElement.FeretBox68Box589
ECodedElement.FeretBox68Center589
ECodedElement.FeretBox68CenterX589
ECodedElement.FeretBox68CenterY590
ECodedElement.FeretBox68Height590
ECodedElement.FeretBox68Width590
ECodedElement.GetCentralMoment591
ECodedElement.GetMoment591
ECodedElement.GetNormalizedCentralMoment592
ECodedElement.GravityCenter593
ECodedElement.GravityCenterX593
ECodedElement.GravityCenterY593
ECodedElement.IsCodedElement594
ECodedElement.IsHole594
ECodedElement.IsObject595
ECodedElement.LargestRun595
ECodedElement.LayerIndex595
ECodedElement.LeftLimit596
ECodedElement.MinimumEnclosingRectangle596
ECodedElement.MinimumEnclosingRectangleAngle596
ECodedElement.MinimumEnclosingRectangleCenter597
ECodedElement.MinimumEnclosingRectangleCenterX597
ECodedElement.MinimumEnclosingRectangleCenterY598
ECodedElement.MinimumEnclosingRectangleHeight598
ECodedElement.MinimumEnclosingRectangleWidth598
ECodedElement.RenderMask599

ECodedElement.RightLimit599
ECodedElement.RunCount600
ECodedElement.RunsIterator600
ECodedElement.SigmaX600
ECodedElement.SigmaXX601
ECodedElement.SigmaXY601
ECodedElement.SigmaY601
ECodedElement.SigmaYY602
ECodedElement.TopLimit602
ECodedImage Class603
ECodedImage.AddFeat615
ECodedImage.AddRunToObj615
ECodedImage.AnalyseObjects616
ECodedImage.BlackClass617
ECodedImage.BlinkFeatures617
ECodedImage.BuildHoles617
ECodedImage.BuildLabeledObjects618
ECodedImage.BuildLabeledRuns619
ECodedImage.BuildObjects619
ECodedImage.BuildRuns620
ECodedImage.Connexity621
ECodedImage.Continuous622
ECodedImage.CurrentObjPtr622
ECodedImage.CurrentRunPtr622
ECodedImage.DetachRunsFromObj623
ECodedImage.DrawDiagonals623
ECodedImage.DrawObject624
ECodedImage.DrawObjectFeature626
ECodedImage.DrawObjectFeatureWithCurrentPen628
ECodedImage.DrawObjects629
ECodedImage.DrawObjectsFeature631
ECodedImage.DrawObjectsFeatureWithCurrentPen632
ECodedImage.DrawObjectsWithCurrentPen634
ECodedImage.DrawObjectWithCurrentPen635
ECodedImage.ECodedImage636
ECodedImage.FeatureAverage636
ECodedImage.FeatureDeviation637
ECodedImage.FeatureMaximum637
ECodedImage.FeatureMinimum638
ECodedImage.FeatureVariance638
ECodedImage.FirstObjPtr639
ECodedImage.GetCurrentObjData639
ECodedImage.GetCurrentRunData640
ECodedImage.GetFeatData640
ECodedImage.GetFeatDataSize641
ECodedImage.GetFeatDataType642
ECodedImage.GetFeatNum642
ECodedImage.GetFeatPtrByNum643
ECodedImage.GetFeatSize643

ECodedImage.GetFirstHole	644
ECodedImage.GetFirstObjData	644
ECodedImage.GetFirstRunData	645
ECodedImage.GetFirstRunPtr	645
ECodedImage.GetHoleParentObject	645
ECodedImage.GetLastObjData	646
ECodedImage.GetLastRunData	646
ECodedImage.GetLastRunPtr	647
ECodedImage.GetNextHole	647
ECodedImage.GetNextObjData	647
ECodedImage.GetNextObjPtr	648
ECodedImage.GetNextRunData	648
ECodedImage.GetNextRunPtr	649
ECodedImage.GetNumHoles	649
ECodedImage.GetNumObjectRuns	650
ECodedImage.GetObjDataPtr	651
ECodedImage.GetObjectData	651
ECodedImage.GetObjectFeature	652
ECodedImage.GetObjFirstRunPtr	654
ECodedImage.GetObjLastRunPtr	655
ECodedImage.GetObjPtr	655
ECodedImage.GetObjPtrByCoordinates	656
ECodedImage.GetObjPtrByPos	656
ECodedImage.GetPreviousObjData	657
ECodedImage.GetPreviousObjPtr	657
ECodedImage.GetPreviousRunData	657
ECodedImage.GetPreviousRunPtr	658
ECodedImage.GetRunData	658
ECodedImage.GetRunDataPtr	659
ECodedImage.GetRunPtr	660
ECodedImage.GetRunPtrByCoordinates	660
ECodedImage.HighColorThreshold	661
ECodedImage.HighImage	661
ECodedImage.HighThreshold	661
ECodedImage.IsHole	662
ECodedImage.IsObjectSelected	662
ECodedImage.LastObjPtr	663
ECodedImage.LimitAngle	663
ECodedImage.LowColorThreshold	664
ECodedImage.LowImage	664
ECodedImage.LowThreshold	664
ECodedImage.MaxObjects	665
ECodedImage.NeutralClass	665
ECodedImage.NumFeatures	666
ECodedImage.NumHoleRuns	666
ECodedImage.NumObjects	666
ECodedImage.NumRuns	667
ECodedImage.NumSelectedObjects	667
ECodedImage.ObjectConvexHull	668

ECodedImage.RemoveAllFeats	668
ECodedImage.RemoveAllObjects	668
ECodedImage.RemoveAllRuns	669
ECodedImage.RemoveHoles	669
ECodedImage.RemoveObject	670
ECodedImage.RemoveRun	670
ECodedImage.ResetContinuousMode	671
ECodedImage.SelectAllObjects	671
ECodedImage.SelectHoles	671
ECodedImage.SelectObject	672
ECodedImage.SelectObjectsUsingFeature	673
ECodedImage.SelectObjectsUsingPosition	674
ECodedImage.SetFeatInfo	675
ECodedImage.SetFirstRunPtr	675
ECodedImage.SetLastRunPtr	676
ECodedImage.SortObjectsUsingFeature	676
ECodedImage.Threshold	677
ECodedImage.ThresholdImage	677
ECodedImage.TrueThreshold	678
ECodedImage.UnselectAllObjects	678
ECodedImage.UnselectHoles	678
ECodedImage.UnselectObject	679
ECodedImage.WhiteClass	680
 ECodedImage2 Class	680
ECodedImage2.ClearFeatureCache	683
ECodedImage2.Draw	683
ECodedImage2.DrawLine	688
ECodedImage2.DrawLineWithCurrentPen	693
ECodedImage2.DrawHole	695
ECodedImage2.DrawLineFeature	697
ECodedImage2.DrawLineFeatureWithCurrentPen	700
ECodedImage2.DrawLineWithCurrentPen	702
ECodedImage2.DrawObject	703
ECodedImage2.DrawObjectFeature	706
ECodedImage2.DrawLineObjectFeatureWithCurrentPen	708
ECodedImage2.DrawLineObjectWithCurrentPen	710
ECodedImage2.DrawWithCurrentPen	711
ECodedImage2.ECodedImage2	713
ECodedImage2.FindObject	714
ECodedImage2.GetObj	714
ECodedImage2.GetObjCount	715
ECodedImage2.GetParentObject	716
ECodedImage2.Height	716
ECodedImage2.LayerCount	717
ECodedImage2.RenderMask	717
ECodedImage2.StartY	718
ECodedImage2.Width	719
 EColorLookup Class	719

EColorLookup.AdjustGainOffset	721
EColorLookup.Calibrate	722
EColorLookup.ColorSystemIn	724
EColorLookup.ColorSystemOut	725
EColorLookup.ConvertFromRgb	725
EColorLookup.ConvertToRgb	726
EColorLookup.EColorLookup	726
EColorLookup.IndexBits	727
EColorLookup.Interpolation	727
EColorLookup.Transform	728
EColorLookup.WhiteBalance	729
EColorRangeThresholdSegmenter Class	730
EColorRangeThresholdSegmenter.HighThreshold	731
EColorRangeThresholdSegmenter.LowThreshold	731
EColorSingleThresholdSegmenter Class	731
EColorSingleThresholdSegmenter.Threshold	732
EColorVector Class	732
EColorVector.AddElement	733
EColorVector.EColorVector	734
EColorVector.GetElement	734
EColorVector.operator[]	735
EColorVector.operator=	735
EColorVector.RawDataPtr	736
EColorVector.SetElement	736
EDepthMap Class	737
EDepthMap16 Class	737
EDepthMap16.AsElImageBW16	738
EDepthMap16.EDepthMap16	738
EDepthMap16.GetPixel	739
EDepthMap16.Precision	740
EDepthMap16.RootROI	740
EDepthMap16.Serialize	740
EDepthMap16.SetPixel	741
EDepthMap16.Type	741
EDepthMap16.UndefinedValue	742
EDepthMap32f Class	742
EDepthMap32f.EDepthMap32f	743
EDepthMap32f.GetPixel	744
EDepthMap32f.RootROI	744
EDepthMap32f.Serialize	744
EDepthMap32f.SetPixel	745
EDepthMap32f.Type	745
EDepthMap32f.UndefinedValue	746
EDepthMap8 Class	746
EDepthMap8.AsElImageBW8	747
EDepthMap8.EDepthMap8	747
EDepthMap8.GetPixel	748

EDepthMap8.RootROI	749
EDepthMap8.Serialize	749
EDepthMap8.SetPixel	749
EDepthMap8.Type	750
EDepthMap8.UndefinedValue	750
EDepthMapROI Class	751
EDepthMapROI16 Class	751
EDepthMapROI16.AsEROIBW16	752
EDepthMapROI16.Attach	752
EDepthMapROI16.CreateNew	753
EDepthMapROI16.EDepthMapROI16	753
EDepthMapROI16.GetPixel	754
EDepthMapROI16.Precision	754
EDepthMapROI16.RootROI	755
EDepthMapROI16.Serialize	755
EDepthMapROI16.SetPixel	755
EDepthMapROI16.Type	756
EDepthMapROI16.UndefinedValue	756
EDepthMapROI32f Class	757
EDepthMapROI32f.Attach	758
EDepthMapROI32f.CreateNew	758
EDepthMapROI32f.EDepthMapROI32f	759
EDepthMapROI32f.GetPixel	759
EDepthMapROI32f.RootROI	760
EDepthMapROI32f.Serialize	760
EDepthMapROI32f.SetPixel	760
EDepthMapROI32f.Type	761
EDepthMapROI32f.UndefinedValue	761
EDepthMapROI8 Class	762
EDepthMapROI8.AsEROIBW8	763
EDepthMapROI8.Attach	763
EDepthMapROI8.CreateNew	764
EDepthMapROI8.EDepthMapROI8	764
EDepthMapROI8.GetPixel	765
EDepthMapROI8.RootROI	765
EDepthMapROI8.Serialize	765
EDepthMapROI8.SetPixel	766
EDepthMapROI8.Type	766
EDepthMapROI8.UndefinedValue	767
EException Class	767
EException.EException	768
EException.Error	769
EException.operator=	769
EException.What	769
EFilePointerSerializer Class	770
EFilePointerSerializer.Close	770
EFilePointerSerializer.Writing	771

EFileSerializer Class	771
EFileSerializer.Close	771
EFileSerializer.Writing	772
EFloatRange Class	772
EFloatRange.EFloatRange	773
EFloatRange.IsInRange	773
EFloatRange.LowerBound	774
EFloatRange.SetBounds	774
EFloatRange.SetFromBaseAndTolerance	775
EFloatRange.UpperBound	775
EFoundPattern Class	776
EFoundPattern.Angle	778
EFoundPattern.Center	778
EFoundPattern.Draw	778
EFoundPattern.DrawBoundingBox	780
EFoundPattern.DrawCenter	780
EFoundPattern.DrawFeaturePoints	781
EFoundPattern.DrawWithCurrentPen	781
EFoundPattern.EFoundPattern	782
EFoundPattern.operator!=	782
EFoundPattern.operator=	783
EFoundPattern.operator==	783
EFoundPattern.Quadrangle	784
EFoundPattern.Scale	784
EFoundPattern.Score	784
EFrame Class	785
EFrame.Angle	786
EFrame.CenterX	786
EFrame.CenterY	787
EFrame.CopyTo	787
EFrame.EFrame	788
EFrame.GlobalToLocal	789
EFrame.LocalToGlobal	789
EFrame.operator=	790
EFrame.Scale	790
EFrameShape Class	790
EFrameShape.Angle	793
EFrameShape.Center	793
EFrameShape.CenterX	793
EFrameShape.CenterY	794
EFrameShape.Closest	794
EFrameShape.CopyTo	794
EFrameShape.Drag	795
EFrameShape.Draw	795
EFrameShape.DrawWithCurrentPen	796
EFrameShape.EFrameShape	797
EFrameShape.HitTest	797
EFrameShape.operator=	798

EFrameShape.Scale	798
EFrameShape.Set	799
EFrameShape.SetCenterXY	799
EFrameShape.SetSize	800
EFrameShape.SizeX	800
EFrameShape.SizeY	801
EFrameShape.Type	801
EGrayscaleDoubleThresholdSegmenter Class	802
EGrayscaleDoubleThresholdSegmenter.HighThreshold	802
EGrayscaleDoubleThresholdSegmenter.LowThreshold	803
EGrayscaleSingleThresholdSegmenter Class	803
EGrayscaleSingleThresholdSegmenter.AbsoluteThreshold	804
EGrayscaleSingleThresholdSegmenter.IsFirstApplication	805
EGrayscaleSingleThresholdSegmenter.LastThreshold	805
EGrayscaleSingleThresholdSegmenter.Mode	806
EGrayscaleSingleThresholdSegmenter.RelativeThreshold	806
EHarrisCornerDetector Class	806
EHarrisCornerDetector.Apply	808
EHarrisCornerDetector.DerivationScale	808
EHarrisCornerDetector.EHarrisCornerDetector	809
EHarrisCornerDetector.GradientNormalizationEnabled	809
EHarrisCornerDetector.IntegrationScale	810
EHarrisCornerDetector.SubpixelPrecisionEnabled	810
EHarrisCornerDetector.Threshold	810
EHarrisCornerDetector.ThresholdingMode	811
EHarrisInterestPoints Class	811
EHarrisInterestPoints.Draw	813
EHarrisInterestPoints.DrawCorner	814
EHarrisInterestPoints.DrawCornerWithCurrentPen	816
EHarrisInterestPoints.DrawWithCurrentPen	816
EHarrisInterestPoints.EHarrisInterestPoints	817
EHarrisInterestPoints.GetCornerness	818
EHarrisInterestPoints.GetGradientMagnitude	818
EHarrisInterestPoints.GetGradientOrientation	819
EHarrisInterestPoints.GetGradientX	819
EHarrisInterestPoints.GetGradientY	819
EHarrisInterestPoints.GetPoint	820
EHarrisInterestPoints.GetX	820
EHarrisInterestPoints.GetY	821
EHarrisInterestPoints.PointCount	821
EHDRCColorFuser Class	821
EHDRCColorFuser.EHDRCColorFuser	822
EHDRCColorFuser.Fuse	823
EHDRCColorFuser.GetFusedImage	823
EHDRFuser Class	824
EHDRFuser.EHDRFuser	824
EHDRFuser.Fuse	825

EHDRFuser.GetFusedImage	825
EHitAndMissKernel Class	826
EHitAndMissKernel.EHitAndMissKernel	827
EHitAndMissKernel.EndX	829
EHitAndMissKernel.EndY	829
EHitAndMissKernel.GetValue	829
EHitAndMissKernel.GetSize	830
EHitAndMissKernel.SetValue	831
EHitAndMissKernel.StartX	832
EHitAndMissKernel.StartY	832
EHole Class	832
EHole.ParentObjectIndex	833
EImageBW1 Class	833
EImageBW1.EImageBW1	834
EImageBW1.GetBitIndex	834
EImageBW1.operator=	835
EImageBW16 Class	836
EImageBW16.EImageBW16	836
EImageBW16.operator=	837
EImageBW32 Class	837
EImageBW32.EImageBW32	838
EImageBW32.operator=	839
EImageBW8 Class	839
EImageBW8.EImageBW8	840
EImageBW8.operator=	841
EImageC15 Class	841
EImageC15.EImageC15	842
EImageC15.operator=	842
EImageC16 Class	843
EImageC16.EImageC16	843
EImageC16.operator=	844
EImageC24 Class	845
EImageC24.EImageC24	845
EImageC24.operator=	846
EImageC24A Class	847
EImageC24A.EImageC24A	847
EImageC24A.operator=	848
EImageC48 Class	849
EImageC48.EImageC48	849
EImageC48.operator=	850
EImageEncoder Class	850
EImageEncoder.BinaryImageSegmenter	853
EImageEncoder.ColorRangeThresholdSegmenter	853
EImageEncoder.ColorSingleThresholdSegmenter	854
EImageEncoder.ContinuousModeEnabled	854
EImageEncoder.ContinuousModeMaxHeight	855

EImageEncoder.EImageEncoder	855
EImageEncoder.Encode	856
EImageEncoder.EncodingConnexity	857
EImageEncoder.FlushContinuousMode	858
EImageEncoder.GrayscaleDoubleThresholdSegmenter	858
EImageEncoder.GrayscaleSingleThresholdSegmenter	859
EImageEncoder.ImageRangeSegmenter	859
EImageEncoder.LabeledImageSegmenter	859
EImageEncoder.ReferenceImageSegmenter	860
EImageEncoder.ResetContinuousMode	860
EImageEncoder.SegmentationMethod	860
EImageRangeSegmenter Class	861
EImageRangeSegmenter.BlackLayerEncoded	862
EImageRangeSegmenter.BlackLayerIndex	863
EImageRangeSegmenter.HighImageBW16	863
EImageRangeSegmenter.HighImageBW8	863
EImageRangeSegmenter.HighImageC24	864
EImageRangeSegmenter.LowImageBW16	864
EImageRangeSegmenter.LowImageBW8	864
EImageRangeSegmenter.LowImageC24	865
EImageRangeSegmenter.WhiteLayerEncoded	865
EImageRangeSegmenter.WhiteLayerIndex	865
EImageSegmenter Class	866
EKernel Class	866
EKernel.EKernel	868
EKernel.Gain	869
EKernel.GetKernelData	869
EKernel.Offset	870
EKernel.OutsideValue	870
EKernel.RawDataPtr	870
EKernel.Rectifier	871
EKernel.SetKernelData	871
EKernel.SetSize	877
EKernel.SizeX	877
EKernel.SizeY	877
ELabeledImageSegmenter Class	878
ELabeledImageSegmenter.MaxLayer	878
ELabeledImageSegmenter.MinLayer	879
ELandmark Class	879
ELandmark.ELandmark	880
ELandmark.operator=	880
ELandmark.SensorX	881
ELandmark.SensorY	881
ELandmark.WorldX	881
ELandmark.WorldY	882
ELaserLineExtractor Class	882
ELaserLineExtractor.AnalysisMode	883

ELaserLineExtractor.AnalysisThreshold	884
ELaserLineExtractor.DepthMap	884
ELaserLineExtractor.ELaserLineExtractor	884
ELaserLineExtractor.EnableSmoothing	885
ELaserLineExtractor.ExtractProfileFromFrame	885
ELaserLineExtractor.Profile	886
ELaserLineExtractor.SetSmoothingParameters	886
ELine Class	887
ELine.CopyTo	888
ELine.ELine	889
ELine.End	890
ELine.GetAngleBetweenLines	890
ELine.GetDistanceBetweenPointAndLine	891
ELine.GetIntersectionOfLines	891
ELine.GetPoint	892
ELine.GetProjectionOfPointOnLine	893
ELine.Length	893
ELine.operator=	894
ELine.Org	894
ELine.Serialize	894
ELine.SetFromOriginAndEnd	895
ELine.SetFromTwoPoints	895
ELineGauge Class	896
ELineGauge.Active	901
ELineGauge.AddSkipRange	902
ELineGauge.AverageDistance	903
ELineGauge.ClippingMode	903
ELineGauge.CopyTo	903
ELineGauge.Drag	904
ELineGauge.Draw	905
ELineGauge.DrawWithCurrentPen	906
ELineGauge.ELineGauge	906
ELineGauge.FilteringThreshold	907
ELineGauge.GetMeasuredPeak	907
ELineGauge.GetMeasuredPoint	908
ELineGauge.GetMinNumFitSamples	909
ELineGauge.GetSample	909
ELineGauge.GetSkipRange	910
ELineGauge.HitTest	911
ELineGauge.HVConstraint	911
ELineGauge.KnownAngle	911
ELineGauge.Line	912
ELineGauge.Measure	912
ELineGauge.MeasuredLine	913
ELineGauge.MeasureSample	913
ELineGauge.MeasureWithoutFitting	914
ELineGauge.MinAmplitude	914
ELineGauge.MinArea	915

ELineGauge.NumFilteringPasses	915
ELineGauge.NumMeasuredPoints	915
ELineGauge.NumSamples	916
ELineGauge.NumSkipRanges	916
ELineGauge.NumValidSamples	916
ELineGauge.operator=	917
ELineGauge.Plot	917
ELineGauge.PlotWithCurrentPen	919
ELineGauge.Process	920
ELineGauge.RectangularSamplingArea	920
ELineGauge.RemoveAllSkipRanges	921
ELineGauge.RemoveSkipRange	921
ELineGauge.SamplingStep	921
ELineGauge.SetMinNumFitSamples	922
ELineGauge.Shape	923
ELineGauge.Smoothing	923
ELineGauge.Thickness	923
ELineGauge.Threshold	924
ELineGauge.Tolerance	924
ELineGauge.TransitionChoice	924
ELineGauge.TransitionIndex	925
ELineGauge.TransitionType	925
ELineGauge.Type	925
ELineGauge.Valid	926
ELineShape Class	926
ELineShape.Angle	928
ELineShape.Center	929
ELineShape.CenterX	929
ELineShape.CenterY	929
ELineShape.Closest	930
ELineShape.CopyTo	930
ELineShape.Drag	931
ELineShape.Draw	931
ELineShape.DrawWithCurrentPen	932
ELineShape.End	933
ELineShape.GetPoint	933
ELineShape.HitTest	933
ELineShape.Length	934
ELineShape.Line	934
ELineShape.operator=	934
ELineShape.Org	935
ELineShape.Scale	935
ELineShape.SetCenterXY	935
ELineShape.SetFromOriginAndEnd	936
ELineShape.SetFromTwoPoints	936
ELineShape.Type	937
EListItem Class	937
EMatcher Class	938

EMatcher.AngleStep	943
EMatcher.ClearImage	944
EMatcher.ContrastMode	944
EMatcher.CopyLearnedPattern	944
EMatcher.CopyTo	945
EMatcher.CorrelationMode	945
EMatcher.DontCareThreshold	946
EMatcher.DrawPosition	946
EMatcher.DrawPositions	948
EMatcher.DrawPositionsWithCurrentPen	949
EMatcher.DrawPositionWithCurrentPen	950
EMatcher.EMatcher	951
EMatcher.FilteringMode	952
EMatcher.FinalReduction	953
EMatcher.GetPixelDimensions	953
EMatcherGetPosition	954
EMatcher.InitialMinScore	954
EMatcher.Interpolate	955
EMatcher.IsotropicScale	955
EMatcher.LearnPattern	956
EMatcher.Load	956
EMatcher.Match	957
EMatcher.MaxAngle	958
EMatcher.MaxInitialPositions	958
EMatcher.MaxPositions	959
EMatcher.MaxScale	959
EMatcher.MaxScaleX	960
EMatcher.MaxScaleY	960
EMatcher.MinAngle	961
EMatcher.MinReducedArea	961
EMatcher.MinScale	962
EMatcher.MinScaleX	962
EMatcher.MinScaleY	963
EMatcher.MinScore	963
EMatcher.NumPositions	964
EMatcher.NumReductions	964
EMatcher.operator=	964
EMatcher.PatternHeight	965
EMatcher.PatternLearned	965
EMatcher.PatternType	965
EMatcher.PatternWidth	966
EMatcher.Save	966
EMatcher.ScaleStep	967
EMatcher.ScaleXStep	967
EMatcher.ScaleYStep	967
EMatcher.SetExtension	968
EMatcher.SetPixelDimensions	968
EMatcher.Version	969
EMatrixCode Class	969

EMatrixCode.Angle975
EMatrixCode.AxialNonUniformity976
EMatrixCode.AxialNonUniformityGrade976
EMatrixCode.CellDefects976
EMatrixCode.Center977
EMatrixCode.Contrast977
EMatrixCode.ContrastGrade977
EMatrixCode.ContrastType978
EMatrixCode.DataMatrixCellHeight978
EMatrixCode.DataMatrixCellWidth979
EMatrixCode.DecodedString979
EMatrixCode.Draw979
EMatrixCode.DrawErrors981
EMatrixCode.DrawErrorsWithCurrentPen982
EMatrixCode.DrawWithCurrentPen983
EMatrixCode.EMatrixCode983
EMatrixCode.Family984
EMatrixCode.FinderPatternDefects984
EMatrixCode.Flipping985
EMatrixCode.Found985
EMatrixCode.GetCorner986
EMatrixCode.GetDecodedDataElement986
EMatrixCode.HorizontalMarkGrowth987
EMatrixCode.HorizontalMarkMisplacement987
EMatrixCode.IsGS1987
EMatrixCode.Iso15415GradingParameters988
EMatrixCode.Iso29158GradingParameters988
EMatrixCode.Load989
EMatrixCode.LocationThreshold989
EMatrixCode.LogicalSize989
EMatrixCode.LogicalSizeHeight990
EMatrixCode.LogicalSizeWidth990
EMatrixCode.MeasuredPrintGrowth990
EMatrixCode.NumErrors991
EMatrixCode.operator=991
EMatrixCode.OverallGrade992
EMatrixCode.PrintGrowth992
EMatrixCode.PrintGrowthGrade993
EMatrixCode.ReadingThreshold993
EMatrixCode.Save993
EMatrixCode.SemiT10GradingParameters994
EMatrixCode.SetCorner994
EMatrixCode.SymbolContrastSNR995
EMatrixCode.UnusedErrorCorrection995
EMatrixCode.UnusedErrorCorrectionGrade996
EMatrixCode.VerticalMarkGrowth996
EMatrixCode.VerticalMarkMisplacement997
EMatrixCodeReader Class997
EMatrixCodeReader.ComputeGrading999

EMatrixCodeReader.EMatrixCodeReader	1000
EMatrixCodeReader.GetLearnMaskElement	1000
EMatrixCodeReader.Learn	1001
EMatrixCodeReader.LearnMore	1001
EMatrixCodeReader.Load	1002
EMatrixCodeReader.MaxHeightWidthRatio	1003
EMatrixCodeReader.MaximumPrintGrowth	1003
EMatrixCodeReader.MinimumPrintGrowth	1004
EMatrixCodeReader.NominalPrintGrowth	1004
EMatrixCodeReader.Read	1005
EMatrixCodeReader.Reset	1005
EMatrixCodeReader.Save	1006
EMatrixCodeReaderSearchParams	1006
EMatrixCodeReader.SetIso29158CalibrationParameters	1007
EMatrixCodeReader.SetLearnMaskElement	1008
EMatrixCodeReader.TimeOut	1008
EMeasurementUnit Class	1009
EMeasurementUnit.ConversionFactorTo	1010
EMeasurementUnit.EMeasurementUnit	1010
EMeasurementUnit.GetStockMeasurementUnit	1011
EMeasurementUnit.Magnitude	1011
EMeasurementUnit.Name	1012
EMovingAverage Class	1012
EMovingAverage.Average	1013
EMovingAverage.EMovingAverage	1014
EMovingAverage.GetSize	1015
EMovingAverage.Reset	1015
EMovingAverage.SetSize	1016
EMovingAverage.SrcImage	1016
EObject Class	1017
EObject.GetHole	1018
EObject.HoleCount	1018
EObjectRunsIterator Class	1018
EObjectRunsIterator.EndX	1020
EObjectRunsIterator.EObjectRunsIterator	1020
EObjectRunsIterator.First	1021
EObjectRunsIterator.IsDone	1021
EObjectRunsIterator.Length	1021
EObjectRunsIterator.Next	1022
EObjectRunsIterator.operator=	1022
EObjectRunsIterator.StartX	1023
EObjectRunsIterator.Y	1023
EObjectSelection Class	1024
EObjectSelection.Add	1029
EObjectSelection.AddHole	1029
EObjectSelection.AddHoles	1030
EObjectSelection.AddHolesOfSelectedObjects	1031
EObjectSelection.AddLayer	1031

EObjectSelection.AddObject	1032
EObjectSelection.AddObjects	1033
EObjectSelection.AddObjectsUsingFloatFeature	1033
EObjectSelection.AddObjectsUsingIntegerFeature	1035
EObjectSelection.AddObjectsUsingRectangle	1036
EObjectSelection.AddObjectsUsingUnsignedIntegerFeature	1037
EObjectSelection.AddObjectUsingPosition	1039
EObjectSelection.AttachedImage	1039
EObjectSelection.Clear	1040
EObjectSelection.ClearFeatureCache	1040
EObjectSelection.ElementCount	1041
EObjectSelection.EObjectSelection	1041
EObjectSelection.FeatureAverage	1041
EObjectSelection.FeatureDeviation	1042
EObjectSelection.FeatureVariance	1042
EObjectSelection.FeretAngle	1043
EObjectSelection.FloatFeatureMaximum	1043
EObjectSelection.FloatFeatureMinimum	1044
EObjectSelection.GetElement	1044
EObjectSelection.GetFloatFeature	1045
EObjectSelection.GetIndexOfElement	1045
EObjectSelection.GetIntegerFeature	1046
EObjectSelection.GetUnsignedIntegerFeature	1046
EObjectSelection.IntegerFeatureMaximum	1047
EObjectSelection.IntegerFeatureMinimum	1047
EObjectSelection.IsSelected	1047
EObjectSelection.Remove	1049
EObjectSelection.RemoveHole	1049
EObjectSelection.RemoveHoles	1050
EObjectSelection.RemoveLayer	1051
EObjectSelection.RemoveObject	1051
EObjectSelection.RemoveObjectsUsingRectangle	1052
EObjectSelection.RemoveObjectUsingPosition	1053
EObjectSelection.RemoveSelectedHoles	1054
EObjectSelection.RemoveUsingFloatFeature	1054
EObjectSelection.RemoveUsingIntegerFeature	1055
EObjectSelection.RemoveUsingUnsignedIntegerFeature	1056
EObjectSelection.RenderMask	1057
EObjectSelection.Sort	1058
EObjectSelection.UnsignedIntegerFeatureMaximum	1059
EObjectSelection.UnsignedIntegerFeatureMinimum	1059
EOCR Class	1060
EOCR.AddChar	1067
EOCR.AddPatternFromImage	1068
EOCR.AverageFirstCharDistance	1069
EOCR.BuildObjects	1069
EOCR.CharGetDstX	1070
EOCR.CharGetDstY	1070
EOCR.CharGetHeight	1070

EOCR.CharGetOrgX	1071
EOCR.CharGetOrgY	1071
EOCR.CharGetTotalDstX	1072
EOCR.CharGetTotalDstY	1072
EOCR.CharGetTotalOrgX	1073
EOCR.CharGetTotalOrgY	1073
EOCR.CharGetWidth	1074
EOCR.CharSpacing	1074
EOCR.CompareAspectRatio	1075
EOCR.CutLargeChars	1075
EOCR.DrawChar	1075
EOCR.DrawChars	1077
EOCR.DrawCharsWithCurrentPen	1078
EOCR.DrawCharWithCurrentPen	1079
EOCR.EmptyChars	1080
EOCR.EOCR	1080
EOCR.FindAllChars	1081
EOCR.GetConfidenceRatio	1081
EOCR.GetFirstCharCode	1082
EOCR.GetFirstCharDistance	1082
EOCR.GetPatternBitmap	1083
EOCR.GetPatternClass	1083
EOCR.GetPatternCode	1084
EOCR.GetSecondCharCode	1084
EOCR.GetSecondCharDistance	1084
EOCR.HitChar	1085
EOCR.HitChars	1086
EOCR.LearnPattern	1087
EOCR.LearnPatterns	1088
EOCR.Load	1089
EOCR.MatchChar	1089
EOCR.MatchingMode	1090
EOCR.MaxCharHeight	1090
EOCR.MaxCharWidth	1091
EOCR.MinCharHeight	1091
EOCR.MinCharWidth	1092
EOCR.NewFont	1092
EOCR.NoiseArea	1093
EOCR.NumChars	1093
EOCR.NumPatterns	1093
EOCR.operator=	1094
EOCR.PatternHeight	1094
EOCR.PatternWidth	1094
EOCR.ReadText	1095
EOCR.ReadTextWide	1096
EOCR.Recognize	1097
EOCR.RecognizeWide	1098
EOCR.RelativeSpacing	1098
EOCR.RelativeThreshold	1099

EOCR.RemoveBorder	1099
EOCR.RemoveNarrowOrFlat	1100
EOCR.RemovePattern	1100
EOCR.Save	1101
EOCR.SegmentationMode	1101
EOCR.SetPatternClass	1102
EOCR.SetPatternCode	1102
EOCR.ShiftingMode	1103
EOCR.ShiftXTolerance	1103
EOCR.ShiftYTolerance	1104
EOCR.TextColor	1104
EOCR.Threshold	1105
EOCR.TrueThreshold	1105
EOCR2 Class	1106
EOCR2.AddCharactersToDatabase	1112
EOCR2.ChrsHeight	1113
EOCR2.ChrsMaxFragmentation	1113
EOCR2.ChrsSpacingBias	1114
EOCR2.ChrsWidth	1114
EOCR2.ChrsWidthBias	1115
EOCR2.ChrsWidthTolerance	1115
EOCR2.ClearCharacterDatabase	1115
EOCR2.Detect	1116
EOCR2.DetectionDelta	1116
EOCR2.DrawDetection	1117
EOCR2.DrawDetectionWithCurrentPen	1118
EOCR2.DrawRecognition	1119
EOCR2.DrawRecognitionWithCurrentPen	1121
EOCR2.DrawSegmentation	1122
EOCR2.DrawSegmentationWithCurrentPen	1123
EOCR2.EOCR2	1124
EOCR2.HitTestChar	1124
EOCR2.HitTestLine	1125
EOCR2.HitTestText	1126
EOCR2.HitTestWord	1127
EOCR2.Learn	1128
EOCR2.Load	1129
EOCR2.MaxVariation	1130
EOCR2.Read	1130
EOCR2.ReadText	1131
EOCR2.Recognize	1131
EOCR2.Save	1132
EOCR2.SaveCharacterDatabase	1132
EOCR2.TextAngleTolerance	1133
EOCR2.TextBaseAngle	1133
EOCR2.TextPolarity	1133
EOCR2.Topology	1134
EOCR2Char Class	1134

EOCR2Char.Bitmap	1135
EOCR2Char.BoundingBox	1136
EOCR2Char.Candidates	1136
EOCR2Char.EOCR2Char	1136
EOCR2Char.operator=	1137
EOCR2Char.Text	1137
EOCR2Char.TextCode	1138
EOCR2Line Class	1138
EOCR2Line.BoundingBox	1139
EOCR2Line.EOCR2Line	1139
EOCR2Line.operator=	1140
EOCR2Line.Text	1140
EOCR2Line.Words	1140
EOCR2Text Class	1141
EOCR2Text.BoundingBox	1141
EOCR2Text.EOCR2Text	1142
EOCR2Text.Lines	1142
EOCR2Text.operator=	1143
EOCR2Text.Text	1143
EOCR2Word Class	1143
EOCR2Word.BoundingBox	1144
EOCR2Word.Characters	1145
EOCR2Word.EOCR2Word	1145
EOCR2Word.operator=	1145
EOCR2Word.Text	1146
EOCV Class	1146
EOCV.AccurateTextsLocationScores	1157
EOCV.AdjustCharsLocationRanges	1158
EOCV.AdjustCharsQualityRanges	1159
EOCV.AdjustShiftTolerances	1160
EOCV.AdjustTextsLocationRanges	1160
EOCV.AdjustTextsQualityRanges	1161
EOCV.ClearStatistics	1162
EOCV.ComputeDefaultTolerances	1162
EOCV.ContrastAverage	1163
EOCV.ContrastDeviation	1163
EOCV.ContrastTolerance	1164
EOCV.CreateTemplateChars	1164
EOCV.CreateTemplateObjects	1165
EOCV.CreateTemplateTexts	1165
EOCV.DeleteTemplateChars	1166
EOCV.DeleteTemplateObjects	1167
EOCV.DeleteTemplateTexts	1167
EOCV.Diagnostics	1168
EOCV.DrawTemplateChars	1168
EOCV.DrawTemplateCharsWithCurrentPen	1170
EOCV.DrawTemplateObjects	1171
EOCV.DrawTemplateObjectsWithCurrentPen	1172

EOCV.DrawTemplateTexts	1173
EOCV.DrawTemplateTextsChars	1174
EOCV.DrawTemplateTextsCharsWithCurrentPen	1176
EOCV.DrawTemplateTextsWithCurrentPen	1177
EOCV.DrawText	1178
EOCV.DrawTextChars	1179
EOCV.DrawTextCharsWithCurrentPen	1181
EOCV.DrawTexts	1182
EOCV.DrawTextsChars	1183
EOCV.DrawTextsCharsWithCurrentPen	1185
EOCV.DrawTextsWithCurrentPen	1186
EOCV.DrawTextWithCurrentPen	1187
EOCV.EOCV	1188
EOCV.FreeCharCount	1188
EOCV.GatherTextsCharsParameters	1189
EOCV.GatherTextsParameters	1189
EOCV.GetFreeChar	1190
EOCV.GetNumTextChars	1191
EOCV.GetText	1191
EOCV.GetTextCharParameters	1192
EOCV.GetTextCharPoint	1192
EOCV.GetTextParameters	1194
EOCV.GetTextPoint	1195
EOCV.Inspect	1196
EOCV.Learn	1197
EOCV.Load	1197
EOCV.LocationMode	1198
EOCV.NormalizeLocationScore	1198
EOCV.NumTexts	1199
EOCV.ReduceLocationScore	1199
EOCV.ResampleChars	1200
EOCV.SampleBackground	1200
EOCV.SampleContrast	1201
EOCV.SampleForeground	1201
EOCV.SampleThreshold	1202
EOCV.Save	1202
EOCV.ScatterTextsCharsParameters	1203
EOCV.ScatterTextsParameters	1204
EOCV.SelectSampleTexts	1204
EOCV.SelectSampleTextsChars	1205
EOCV.SelectTemplateChars	1206
EOCV.SelectTemplateObjects	1207
EOCV.SelectTemplateTexts	1208
EOCV.SetFreeChar	1209
EOCV.SetText	1210
EOCV.SetTextCharParameters	1211
EOCV.SetTextParameters	1211
EOCV.StatisticsCount	1212
EOCV.TemplateBackground	1212

EOCV.TemplateContrast1213
EOCV.TemplateForeground1213
EOCV.TemplateImage1214
EOCV.TemplateThreshold1214
EOCV.UpdateStatistics1215
EOCV.UsedQualityIndicators1215
EOCV.WhiteOnBlack1216
EOCVChar Class1216
EOCVChar.BackgroundAreaAverage1222
EOCVChar.BackgroundAreaDeviation1222
EOCVChar.BackgroundAreaTolerance1223
EOCVChar.BackgroundSumAverage1223
EOCVChar.BackgroundSumDeviation1223
EOCVChar.BackgroundSumTolerance1224
EOCVChar.Correlation1224
EOCVChar.CorrelationAverage1224
EOCVChar.CorrelationDeviation1225
EOCVChar.CorrelationTolerance1225
EOCVChar.Diagnostics1225
EOCVChar.EOCVChar1226
EOCVChar.ForegroundAreaAverage1226
EOCVChar.ForegroundAreaDeviation1227
EOCVChar.ForegroundAreaTolerance1227
EOCVChar.ForegroundSumAverage1227
EOCVChar.ForegroundSumDeviation1228
EOCVChar.ForegroundSumTolerance1228
EOCVChar.LocationScoreAverage1228
EOCVChar.LocationScoreDeviation1229
EOCVChar.LocationScoreTolerance1229
EOCVChar.MarginWidth1230
EOCVChar.NumContourPoints1230
EOCVChar.operator=1230
EOCVChar.ResetParameters1231
EOCVChar.SampleBackgroundArea1231
EOCVChar.SampleBackgroundSum1232
EOCVChar.SampleForegroundArea1232
EOCVChar.SampleForegroundSum1232
EOCVChar.SampleLocationScore1233
EOCVChar.Selected1233
EOCVChar.ShiftX1233
EOCVChar.ShiftXAverage1234
EOCVChar.ShiftXBias1234
EOCVChar.ShiftXDeviation1234
EOCVChar.ShiftXMax1235
EOCVChar.ShiftXMin1235
EOCVChar.ShiftXStride1235
EOCVChar.ShiftXTolerance1236
EOCVChar.ShiftY1236
EOCVChar.ShiftYAverage1236

EOCVChar.ShiftYBias	1237
EOCVChar.ShiftYDeviation	1237
EOCVChar.ShiftYMax	1237
EOCVChar.ShiftYMin	1238
EOCVChar.ShiftYStride	1238
EOCVChar.ShiftYTolerance	1238
EOCVChar.TemplateBackgroundArea	1239
EOCVChar.TemplateBackgroundSum	1239
EOCVChar.TemplateForegroundArea	1239
EOCVChar.TemplateForegroundSum	1240
EOCVChar.TemplateLocationScore	1240
EOCVChar.WhiteOnBlack	1241
 EOCVText Class	1241
EOCVText.BackgroundAreaAverage	1249
EOCVText.BackgroundAreaDeviation	1250
EOCVText.BackgroundAreaTolerance	1250
EOCVText.BackgroundSumAverage	1250
EOCVText.BackgroundSumDeviation	1251
EOCVText.BackgroundSumTolerance	1251
EOCVText.Correlation	1251
EOCVText.CorrelationAverage	1252
EOCVText.CorrelationDeviation	1252
EOCVText.CorrelationTolerance	1252
EOCVText.Diagnostics	1253
EOCVText.EOCVText	1253
EOCVText.ForegroundAreaAverage	1254
EOCVText.ForegroundAreaDeviation	1254
EOCVText.ForegroundAreaTolerance	1254
EOCVText.ForegroundSumAverage	1255
EOCVText.ForegroundSumDeviation	1255
EOCVText.ForegroundSumTolerance	1255
EOCVText.IsotropicScaling	1256
EOCVText.LocationScoreAverage	1256
EOCVText.LocationScoreDeviation	1256
EOCVText.LocationScoreTolerance	1257
EOCVText.MarginWidth	1257
EOCVText.NumContourPoints	1257
EOCVText.operator=	1258
EOCVText.ResetParameters	1258
EOCVText.SampleBackgroundArea	1259
EOCVText.SampleBackgroundSum	1259
EOCVText.SampleForegroundArea	1259
EOCVText.SampleForegroundSum	1260
EOCVText.SampleLocationScore	1260
EOCVText.ScaleX	1260
EOCVText.ScaleXAverage	1261
EOCVText.ScaleXBias	1261
EOCVText.ScaleXCount	1261
EOCVText.ScaleXDeviation	1262

EOCVText.ScaleXMax	1262
EOCVText.ScaleXMin	1262
EOCVText.ScaleXTolerance	1263
EOCVText.ScaleY	1263
EOCVText.ScaleYAverage	1263
EOCVText.ScaleYBias	1264
EOCVText.ScaleYCount	1264
EOCVText.ScaleYDeviation	1264
EOCVText.ScaleYMax	1265
EOCVText.ScaleYMin	1265
EOCVText.ScaleYTolerance	1265
EOCVText.Selected	1266
EOCVText.Shear	1266
EOCVText.ShearAverage	1266
EOCVText.ShearBias	1267
EOCVText.ShearCount	1267
EOCVText.ShearDeviation	1267
EOCVText.ShearMax	1268
EOCVText.ShearMin	1268
EOCVText.ShearTolerance	1268
EOCVText.ShiftX	1269
EOCVText.ShiftXAverage	1269
EOCVText.ShiftXBias	1269
EOCVText.ShiftXDeviation	1270
EOCVText.ShiftXMax	1270
EOCVText.ShiftXMin	1270
EOCVText.ShiftXStride	1271
EOCVText.ShiftXTolerance	1271
EOCVText.ShiftY	1271
EOCVText.ShiftYAverage	1272
EOCVText.ShiftYBias	1272
EOCVText.ShiftYDeviation	1272
EOCVText.ShiftYMax	1273
EOCVText.ShiftYMin	1273
EOCVText.ShiftYStride	1273
EOCVText.ShiftYTolerance	1274
EOCVText.Skew	1274
EOCVText.SkewAverage	1274
EOCVText.SkewBias	1275
EOCVText.SkewCount	1275
EOCVText.SkewDeviation	1275
EOCVText.SkewMax	1276
EOCVText.SkewMin	1276
EOCVText.SkewTolerance	1276
EOCVText.TemplateBackgroundArea	1277
EOCVText.TemplateBackgroundSum	1277
EOCVText.TemplateForegroundArea	1277
EOCVText.TemplateForegroundSum	1278
EOCVText.TemplateLocationScore	1278

EPathVector Class	1278
EPathVector.AddElement	1280
EPathVector.Closed	1280
EPathVector.Draw	1280
EPathVector.DrawWithCurrentPen	1282
EPathVector.EPathVector	1283
EPathVector.GetElement	1283
EPathVector.operator[]	1284
EPathVector.operator=	1284
EPathVector.RawDataPtr	1285
EPathVector.SetElement	1285
EPatternFinder Class	1286
EPatternFinder.Angle	1290
EPatternFinder.AngleBias	1290
EPatternFinder.AngleSearchExtent	1291
EPatternFinder.AngleTolerance	1291
EPatternFinder.AutoTransitionThickness	1291
EPatternFinder.CachedPivot	1292
EPatternFinder.ContrastMode	1292
EPatternFinder.CopyLearnedPattern	1293
EPatternFinder.DrawModel	1293
EPatternFinder.DrawModelWithCurrentPen	1294
EPatternFinder.EPatternFinder	1295
EPatternFinder.Find	1296
EPatternFinder.FindExtension	1296
EPatternFinder.ForcedThreshold	1297
EPatternFinder.Interpolate	1297
EPatternFinder.Learn	1298
EPatternFinder.LearningDone	1298
EPatternFinder.LightBalance	1299
EPatternFinder.LocalSearchMode	1300
EPatternFinder.MaxFeaturePoints	1301
EPatternFinder.MaxInstances	1301
EPatternFinder.MinFeaturePoints	1302
EPatternFinder.MinScore	1302
EPatternFinder.operator=	1302
EPatternFinder.PatternType	1303
EPatternFinder.Pivot	1303
EPatternFinder.ReductionMode	1304
EPatternFinder.ReductionStrength	1304
EPatternFinder.Scale	1305
EPatternFinder.ScaleBias	1305
EPatternFinder.ScaleSearchExtent	1306
EPatternFinder.ScaleTolerance	1306
EPatternFinder.Score	1306
EPatternFinder.ThinStructureMode	1307
EPatternFinder.TransitionThickness	1307
EPatternFinder.Type	1308
EPatternFinder.XSearchExtent	1308

EPatternFinder.YSearchExtent	1309
EPeakVector Class	1309
EPeakVector.AddElement	1310
EPeakVector.EPeakVector	1310
EPeakVector.GetElement	1311
EPeakVector.operator[]	1312
EPeakVector.operator=	1312
EPeakVector.RawDataPtr	1312
EPeakVector.SetElement	1313
EPixelAccessor Class	1313
EPixelAccessor.GetBufferPtr	1314
EPixelAccessor.GetCheckedBufferPtr	1315
EPixelAccessor.Height	1316
EPixelAccessor.IsSameType	1316
EPixelAccessor.IsVoid	1316
EPixelAccessor.RowPitch	1317
EPixelAccessor.Type	1317
EPixelAccessor.Width	1317
EPixelContainer Class	1318
EPixelContainer.AsEBaseROI	1320
EPixelContainer.BitsPerPixel	1320
EPixelContainer.CopyTo	1320
EPixelContainer.Draw	1321
EPixelContainer.GetBufferPtr	1323
EPixelContainer.GetCheckedBufferPtr	1324
EPixelContainer.Height	1324
EPixelContainer.Load	1325
EPixelContainer.operator=	1325
EPixelContainer.PlanesPerPixel	1326
EPixelContainer.RowPitch	1326
EPixelContainer.Save	1326
EPixelContainer.SaveJpeg	1327
EPixelContainer.SaveJpeg2K	1328
EPixelContainer.SetBufferPtr	1328
EPixelContainer.SetSize	1329
EPixelContainer.Width	1330
EPixelRegion Class	1330
EPoint Class	1330
EPoint.Area	1333
EPoint.Argument	1334
EPoint.Center	1334
EPoint.CopyTo	1335
EPoint.Distance	1335
EPoint.Dot	1336
EPoint.EPoint	1336
EPoint.MidPoint	1337
EPoint.Modulus	1338
EPoint.operator-	1338

EPoint.operator!=	1338
EPoint.operator*	1339
EPoint.operator/	1339
EPoint.operator+	1340
EPoint.operator=	1340
EPoint.operator==	1341
EPoint.Project	1341
EPoint.Rotate	1342
EPoint.SetCenterXY	1342
EPoint.Square	1343
EPoint.SquaredDistance	1343
EPoint.X	1344
EPoint.Y	1344
EPointGauge Class	1345
EPointGauge.Active	1349
EPointGauge.Center	1349
EPointGauge.CopyTo	1350
EPointGauge.Drag	1350
EPointGauge.Draw	1351
EPointGauge.DrawWithCurrentPen	1352
EPointGauge.EPointGauge	1352
EPointGauge.GetMeasuredPeak	1353
EPointGauge.GetMeasuredPoint	1354
EPointGauge.HitTest	1354
EPointGauge.HVConstraint	1355
EPointGauge.Measure	1355
EPointGauge.MinAmplitude	1356
EPointGauge.MinArea	1356
EPointGauge.NumMeasuredPoints	1357
EPointGauge.operator=	1357
EPointGauge.Plot	1358
EPointGauge.PlotWithCurrentPen	1359
EPointGauge.Process	1360
EPointGauge.RectangularSamplingArea	1360
EPointGauge.SetCenterXY	1361
EPointGauge.SetTolerances	1361
EPointGauge.Shape	1362
EPointGauge.Smoothing	1362
EPointGauge.Thickness	1363
EPointGauge.Threshold	1363
EPointGauge.Tolerance	1364
EPointGauge.ToleranceAngle	1364
EPointGauge.TransitionChoice	1365
EPointGauge.TransitionIndex	1365
EPointGauge.TransitionType	1366
EPointGauge.Type	1366
EPointGauge.Valid	1366
EPointShape Class	1367

EPointShape.Angle	1369
EPointShape.Center	1369
EPointShape.CenterX	1369
EPointShape.CenterY	1370
EPointShape.Closest	1370
EPointShape.CopyTo	1370
EPointShape.Drag	1371
EPointShape.Draw	1371
EPointShape.DrawWithCurrentPen	1372
EPointShape.HitTest	1373
EPointShape.operator!=	1373
EPointShape.operator=	1374
EPointShape.operator==	1374
EPointShape.Scale	1375
EPointShape.SetCenterXY	1375
EPointShape.Type	1376
EPseudoColorLookup Class	1376
EPseudoColorLookup.EPseudoColorLookup	1376
EPseudoColorLookup.SetShading	1377
QRCode Class	1378
QRCode.DecodedStream	1379
QRCode.Draw	1379
QRCode.DrawWithCurrentPen	1380
QRCode.EQRCODE	1381
QRCode.Geometry	1381
QRCode.IsDecodingReliable	1382
QRCode.Level	1382
QRCode.Model	1382
QRCode.operator=	1383
QRCode.UnusedErrorCorrection	1383
QRCode.Version	1383
QRCodeDecodedStream Class	1384
QRCodeDecodedStream.ApplicationIndicator	1385
QRCodeDecodedStream.CodingMode	1385
QRCodeDecodedStream.DecodedStreamParts	1385
QRCodeDecodedStream.EQRCODEDecodedStream	1386
QRCodeDecodedStream.operator=	1386
QRCodeDecodedStream.RawBitstream	1387
QRCodeDecodedStreamPart Class	1387
QRCodeDecodedStreamPart.DecodedData	1388
QRCodeDecodedStreamPart.Encoding	1388
QRCodeDecodedStreamPart.EQRCODEDecodedStreamPart	1388
QRCodeDecodedStreamPart.operator=	1389
QRCodeGeometry Class	1389
QRCodeGeometry.Draw	1390
QRCodeGeometry.DrawWithCurrentPen	1391
QRCodeGeometry.EQRCODEGeometry	1392
QRCodeGeometry.FinderPatternCenters	1392

EQRCodeGeometry.operator=	1393
EQRCodeGeometry.Position	1393
EQRCodeReader Class	1394
EQRCodeReader.CellPolarityConfidenceThreshold	1396
EQRCodeReader.Decode	1396
EQRCodeReader.Detect	1397
EQRCodeReader.DetectionMethod	1397
EQRCodeReader.DetectionTradeOff	1398
EQRCodeReader.EQRCodeReader	1398
EQRCodeReader.FilterOutUnreliablyDecodedQRCodes	1399
EQRCodeReader.ForegroundDetectionThreshold	1399
EQRCodeReader.MaximumVersion	1399
EQRCodeReader.MinimumIsotropy	1400
EQRCodeReader.MinimumScore	1400
EQRCodeReader.MinimumVersion	1401
EQRCodeReader.PerspectiveMode	1401
EQRCodeReader.Read	1402
EQRCodeReader.ScanPrecision	1402
EQRCodeReader.SearchedModels	1402
EQRCodeReader.SearchField	1403
EQRCodeReader.TimeOut	1403
EQQuadrangle Class	1404
EQQuadrangle.Draw	1404
EQQuadrangle.DrawWithCurrentPen	1406
EQQuadrangle.EQuadrangle	1407
EQQuadrangle.GetPoint	1407
EQQuadrangle.operator=	1408
EQQuadrilateral Class	1408
EQQuadrilateral.Corners	1409
EQQuadrilateral.EQuadrilateral	1409
EQQuadrilateral.operator=	1410
ERectangle Class	1410
ERectangle.CopyTo	1412
ERectangle.ERectangle	1413
ERectangle.GetCorners	1414
ERectangle.GetEdges	1414
ERectangle.GetMidEdges	1415
ERectangle.GetPoint	1416
ERectangle.operator=	1416
ERectangle.SetFromOppositeCorners	1417
ERectangle.SetFromOriginMiddleEnd	1417
ERectangle.SetFromThreeCorners	1418
ERectangle.SetFromTwoPoints	1418
ERectangle.GetSize	1419
ERectangle.SizeX	1419
ERectangle.SizeY	1420
ERectangleGauge Class	1420
ERectangleGauge.Active	1427

ERectangleGauge.ActiveEdges	1427
ERectangleGauge.AddSkipRange	1427
ERectangleGauge.AverageDistance	1428
ERectangleGauge.CopyTo	1429
ERectangleGauge.DisableInnerFiltering	1429
ERectangleGauge.Drag	1430
ERectangleGauge.Draw	1430
ERectangleGauge.DrawWithCurrentPen	1431
ERectangleGauge.ERectangleGauge	1432
ERectangleGauge.FilteringThreshold	1432
ERectangleGauge.GetMeasuredPoint	1433
ERectangleGauge.GetMinNumFitSamples	1434
ERectangleGauge.GetSampleX	1434
ERectangleGauge.GetSampleX	1435
ERectangleGauge.GetSampleY	1436
ERectangleGauge.GetSampleY	1437
ERectangleGauge.GetSkipRange	1437
ERectangleGauge.HitTest	1438
ERectangleGauge.HVConstraint	1438
ERectangleGauge.InnerFilteringEnabled	1439
ERectangleGauge.InnerFilteringThreshold	1439
ERectangleGauge.KnownAngle	1440
ERectangleGauge.Measure	1440
ERectangleGauge.MeduredRectangle	1441
ERectangleGauge.MeasureSample	1441
ERectangleGauge.MeasureWithoutFitting	1442
ERectangleGauge.MinAmplitude	1442
ERectangleGauge.MinArea	1443
ERectangleGauge.NumFilteringPasses	1443
ERectangleGauge.NumSamples	1444
ERectangleGauge.NumSamplesX	1444
ERectangleGauge.NumSamplesX	1444
ERectangleGauge.NumSamplesY	1445
ERectangleGauge.NumSamplesY	1445
ERectangleGauge.NumSkipRanges	1445
ERectangleGauge.NumValidSamples	1446
ERectangleGauge.operator=	1446
ERectangleGauge.Plot	1447
ERectangleGauge.PlotWithCurrentPen	1448
ERectangleGauge.Process	1449
ERectangleGauge.RectangularSamplingArea	1450
ERectangleGauge.RemoveAllSkipRanges	1450
ERectangleGauge.RemoveSkipRange	1450
ERectangleGauge.SamplingStep	1451
ERectangleGauge.SetMinNumFitSamples	1451
ERectangleGauge.Shape	1452
ERectangleGauge.Smoothing	1452
ERectangleGauge.Thickness	1453
ERectangleGauge.Threshold	1453

ERectangleGauge.Tolerance	1453
ERectangleGauge.TransitionChoice	1454
ERectangleGauge.TransitionIndex	1454
ERectangleGauge.TransitionType	1454
ERectangleGauge.Type	1455
ERectangleGauge.Valid	1455
ERectangleShape Class	1455
ERectangleShape.Angle	1458
ERectangleShape.Center	1458
ERectangleShape.CenterX	1459
ERectangleShape.CenterY	1459
ERectangleShape.Closest	1459
ERectangleShape.CopyTo	1460
ERectangleShape.Drag	1460
ERectangleShape.Draw	1461
ERectangleShape.DrawWithCurrentPen	1462
ERectangleShape.GetCorners	1462
ERectangleShape.GetEdges	1463
ERectangleShape.GetMidEdges	1463
ERectangleShape.GetPoint	1464
ERectangleShape.HitTest	1465
ERectangleShape.operator=	1465
ERectangleShape.Rectangle	1465
ERectangleShape.Scale	1466
ERectangleShape.SetCenterXY	1466
ERectangleShape.SetFromOppositeCorners	1467
ERectangleShape.SetFromOriginMiddleEnd	1467
ERectangleShape.SetFromThreeCorners	1468
ERectangleShape.SetFromTwoPoints	1468
ERectangleShape.SetSize	1469
ERectangleShape.SizeX	1469
ERectangleShape.SizeY	1470
ERectangleShape.Type	1470
EReferenceImageSegmenter Class	1470
EReferenceImageSegmenter.BlackLayerEncoded	1472
EReferenceImageSegmenter.BlackLayerIndex	1472
EReferenceImageSegmenter.ReferenceImageBW16	1472
EReferenceImageSegmenter.ReferenceImageBW8	1473
EReferenceImageSegmenter.ReferenceImageC24	1473
EReferenceImageSegmenter.WhiteLayerEncoded	1473
EReferenceImageSegmenter.WhiteLayerIndex	1474
EROI Class	1474
EROI.AsEBaseROI	1477
EROI.Attach	1477
EROI.Children	1477
EROI.CopyTo	1478
EROI.CreateNew	1478
EROI.Detach	1478

EROI.Draw	1479
EROI.DrawLineWithCurrentPen	1481
EROI.GetBufferPtr	1482
EROI.GetCheckedBufferPtr	1482
EROI.Height	1483
EROI.IsRoot	1483
EROI.IsVoid	1484
EROI.operator=	1484
EROI.OrgX	1484
EROI.OrgY	1485
EROI.Parent	1485
EROI.PixelContainer	1485
EROI.RowPitch	1486
EROI.Save	1486
EROI.Serialize	1487
EROI.TotalOrgX	1487
EROI.TotalOrgY	1487
EROI.Width	1488
 EROIBW1 Class	1488
EROIBW1.EROIBW1	1489
EROIBW1.FirstSubROI	1490
EROIBW1.GetBitIndex	1490
EROIBW1.GetNextROI	1491
EROIBW1.GetPixel	1491
EROIBW1.NextSiblingROI	1492
EROIBW1.operator=	1492
EROIBW1.Parent	1493
EROIBW1.Serialize	1493
EROIBW1.SetPixel	1493
EROIBW1.TopParent	1494
 EROIBW16 Class	1495
EROIBW16.EROIBW16	1496
EROIBW16.FirstSubROI	1496
EROIBW16.GetNextROI	1497
EROIBW16.GetPixel	1497
EROIBW16.NextSiblingROI	1498
EROIBW16.operator=	1498
EROIBW16.Parent	1499
EROIBW16.Serialize	1499
EROIBW16.SetPixel	1499
EROIBW16.TopParent	1500
 EROIBW32 Class	1501
EROIBW32.EROIBW32	1502
EROIBW32.FirstSubROI	1502
EROIBW32.GetNextROI	1503
EROIBW32.GetPixel	1503
EROIBW32.NextSiblingROI	1504
EROIBW32.operator=	1504

EROIBW32.Parent	1505
EROIBW32.Serialize	1505
EROIBW32.SetPixel	1505
EROIBW32.TopParent	1506
EROIBW8 Class	1507
EROIBW8.EROIBW8	1508
EROIBW8.FirstSubROI	1508
EROIBW8.GetNextROI	1509
EROIBW8.GetPixel	1509
EROIBW8.NextSiblingROI	1510
EROIBW8.operator=	1510
EROIBW8.Parent	1511
EROIBW8.Serialize	1511
EROIBW8.SetPixel	1511
EROIBW8.TopParent	1512
EROIC15 Class	1513
EROIC15.EROIC15	1514
EROIC15.FirstSubROI	1514
EROIC15.GetNextROI	1515
EROIC15.GetPixel	1515
EROIC15.NextSiblingROI	1516
EROIC15.operator=	1516
EROIC15.Parent	1517
EROIC15.Serialize	1517
EROIC15.SetPixel	1517
EROIC15.TopParent	1518
EROIC16 Class	1519
EROIC16.EROIC16	1520
EROIC16.FirstSubROI	1520
EROIC16.GetNextROI	1521
EROIC16.GetPixel	1521
EROIC16.NextSiblingROI	1522
EROIC16.operator=	1522
EROIC16.Parent	1523
EROIC16.Serialize	1523
EROIC16.SetPixel	1523
EROIC16.TopParent	1524
EROIC24 Class	1525
EROIC24.EROIC24	1526
EROIC24.FirstSubROI	1526
EROIC24.GetNextROI	1527
EROIC24.GetPixel	1527
EROIC24.NextSiblingROI	1528
EROIC24.operator=	1528
EROIC24.Parent	1529
EROIC24.Serialize	1529
EROIC24.SetPixel	1529
EROIC24.TopParent	1530

EROIC24A Class	1531
EROIC24A.EROIC24A	1532
EROIC24A.FirstSubROI	1532
EROIC24A.GetNextROI	1533
EROIC24A.GetPixel	1533
EROIC24A.NextSiblingROI	1534
EROIC24A.operator=	1534
EROIC24A.Parent	1535
EROIC24A.Serialize	1535
EROIC24A.SetPixel	1535
EROIC24A.TopParent	1536
EROIC48 Class	1537
EROIC48.EROIC48	1538
EROIC48.FirstSubROI	1538
EROIC48.GetNextROI	1539
EROIC48.GetPixel	1539
EROIC48.NextSiblingROI	1540
EROIC48.operator=	1540
EROIC48.Parent	1541
EROIC48.Serialize	1541
EROIC48.SetPixel	1541
EROIC48.TopParent	1542
ERotatedBoundingBox Class	1543
ERotatedBoundingBox.Angle	1544
ERotatedBoundingBox.Center	1545
ERotatedBoundingBox.CenterX	1545
ERotatedBoundingBox.CenterY	1545
ERotatedBoundingBox.Draw	1546
ERotatedBoundingBox.DrawWithCurrentPen	1547
ERotatedBoundingBox.ERotatedBoundingBox	1548
ERotatedBoundingBox.Height	1549
ERotatedBoundingBox.LocalToGlobalBox	1549
ERotatedBoundingBox.LocalToGlobalPoint	1549
ERotatedBoundingBox.operator=	1550
ERotatedBoundingBox.Quadrangle	1550
ERotatedBoundingBox.Translate	1551
ERotatedBoundingBox.Width	1551
ESearchParamsType Class	1551
ESearchParamsType.AddContrast	1554
ESearchParamsType.AddFamily	1555
ESearchParamsType.AddFlipping	1555
ESearchParamsType.AddLogicalSize	1556
ESearchParamsType.ClearContrast	1556
ESearchParamsType.ClearFamily	1556
ESearchParamsType.ClearFlipping	1557
ESearchParamsType.ClearLogicalSize	1557
ESearchParamsType.ContrastCount	1557
ESearchParamsType.FamilyCount	1558

ESearchParamsType.FlippingCount	1558
ESearchParamsType.GetContrast	1558
ESearchParamsType.GetFamily	1559
ESearchParamsType.GetFlipping	1559
ESearchParamsType.GetLogicalSize	1560
ESearchParamsType.LogicalSizeCount	1560
ESearchParamsType.RemoveContrast	1560
ESearchParamsType.RemoveFamily	1561
ESearchParamsType.RemoveFlipping	1561
ESearchParamsType.RemoveLogicalSize	1562
ESerializer Class	1562
ESerializer.Close	1563
ESerializer.CreateCallbackReader	1564
ESerializer.CreateCallbackWriter	1565
ESerializer.CreateFileReader	1565
ESerializer.Create.FileWriter	1566
ESerializer.Writing	1567
EShape Class	1567
EShape.Active	1573
EShape.ActiveRecursive	1573
EShape.ActualShape	1574
EShape.ActualShapeRecursive	1574
EShape.Attach	1575
EShape.Closest	1575
EShape.ClosestShape	1575
EShape.Detach	1576
EShape.DetachDaughters	1576
EShape.DisableBehaviorFilter	1576
EShape.DisableTypeFilter	1577
EShape.Drag	1577
EShape.Dragable	1578
EShape.DragableRecursive	1578
EShape.Draw	1578
EShape.DrawWithCurrentPen	1579
EShape.EnableBehaviorFilter	1580
EShape.EnableTypeFilter	1581
EShape.GetAllocated	1581
EShape.GetDaughter	1582
EShape.GetDraggingMode	1582
EShape.GetOptionalDraw	1582
EShape.GetShapeNamed	1583
EShape.HitHandle	1583
EShape.HitShape	1583
EShape.HitTest	1584
EShape.InvalidateWorld	1584
EShape.Labeled	1585
EShape.LabeledRecursive	1585
EShape.Load	1585

EShape.LocalToSensor	1586
EShape.Mother	1586
EShape.Name	1587
EShape.NumDaughters	1587
EShape.PanX	1587
EShape.PanY	1588
EShape.Process	1588
EShape.Resizable	1589
EShape.ResizableRecursive	1589
EShape.Rotatable	1589
EShape.RotatableRecursive	1590
EShape.Save	1590
EShape.Selectable	1591
EShape.SelectableRecursive	1591
EShape.Selected	1591
EShape.SelectedRecursive	1592
EShape.SensorToLocal	1592
EShape.SetAllocated	1592
EShape.SetCursor	1593
EShape.SetDraggingMode	1593
EShape.SetOptionalDraw	1594
EShape.SetPan	1594
EShape.SetZoom	1595
EShape.Type	1596
EShape.Visible	1596
EShape.VisibleRecursive	1596
EShape.WorldShape	1597
EShape.ZoomX	1597
EShape.ZoomY	1597
EThreeLayersImageSegmenter Class	1598
EThreeLayersImageSegmenter.BlackLayerEncoded	1599
EThreeLayersImageSegmenter.BlackLayerIndex	1599
EThreeLayersImageSegmenter.NeutralLayerEncoded	1599
EThreeLayersImageSegmenter.NeutralLayerIndex	1600
EThreeLayersImageSegmenter.WhiteLayerEncoded	1600
EThreeLayersImageSegmenter.WhiteLayerIndex	1600
ETwoLayersImageSegmenter Class	1601
ETwoLayersImageSegmenter.BlackLayerEncoded	1602
ETwoLayersImageSegmenter.BlackLayerIndex	1602
ETwoLayersImageSegmenter.WhiteLayerEncoded	1602
ETwoLayersImageSegmenter.WhiteLayerIndex	1603
EUnwarpingLut Class	1603
EUnwarpingLut.EUnwarpingLut	1603
EVector Class	1604
EVector.Empty	1605
EVector.NumElements	1605
EVector.RemoveElement	1605
EVector.Serialize	1606

EWedge Class	1606
EWedge.Amplitude	1610
EWedge.ApexAngle	1611
EWedge.Breadth	1611
EWedge.CopyTo	1612
EWedge.Direct	1612
EWedge.EndAngle	1612
EWedge.EWedge	1613
EWedge.FullBreadth	1615
EWedge.FullCircle	1615
EWedge.GetCorners	1615
EWedge.GetEdges	1616
EWedge.GetInnerPoint	1617
EWedge.GetMidEdges	1617
EWedge.GetOuterPoint	1618
EWedge.GetPoint	1618
EWedge.InnerApex	1619
EWedge.InnerArcLength	1619
EWedge.InnerDiameter	1619
EWedge.InnerEnd	1620
EWedge.InnerOrg	1620
EWedge.InnerRadius	1620
EWedge.operator=	1621
EWedge.OrgAngle	1621
EWedge.OuterApex	1622
EWedge.OuterArcLength	1622
EWedge.OuterDiameter	1623
EWedge.OuterEnd	1623
EWedge.OuterOrg	1624
EWedge.OuterRadius	1624
EWedge.SetDiameters	1624
EWedge.SetFromCenterAndOrigin	1625
EWedge.SetFromOriginMiddleEnd	1626
EWedge.SetFromTwoPoints	1627
EWedge.SetRadii	1627
EWedgeGauge Class	1628
EWedgeGauge.Active	1634
EWedgeGauge.ActiveEdges	1635
EWedgeGauge.AddSkipRange	1635
EWedgeGauge.AverageDistance	1636
EWedgeGauge.CopyTo	1636
EWedgeGauge.Drag	1637
EWedgeGauge.Draw	1637
EWedgeGauge.DrawWithCurrentPen	1638
EWedgeGauge.EWedgeGauge	1639
EWedgeGauge.FilteringThreshold	1640
EWedgeGauge.GetMeasuredPoint	1640
EWedgeGauge.GetMinNumFitSamples	1641
EWedgeGauge.GetSampleA	1641

EWedgeGauge.GetSampleA	1642
EWedgeGauge.GetSampleR	1643
EWedgeGauge.GetSampleR	1644
EWedgeGauge.GetSkipRange	1644
EWedgeGauge.HitTest	1645
EWedgeGauge.HVConstraint	1645
EWedgeGauge.Measure	1646
EWedgeGauge.MeasuredWedge	1646
EWedgeGauge.MeasureSample	1647
EWedgeGauge.MeasureWithoutFitting	1647
EWedgeGauge.MinAmplitude	1648
EWedgeGauge.MinArea	1648
EWedgeGauge.NumFilteringPasses	1648
EWedgeGauge.NumSamples	1649
EWedgeGauge.NumSamplesA	1649
EWedgeGauge.NumSamplesA	1649
EWedgeGauge.NumSamplesR	1650
EWedgeGauge.NumSamplesR	1650
EWedgeGauge.NumSkipRanges	1650
EWedgeGauge.NumValidSamples	1651
EWedgeGauge.operator=	1651
EWedgeGauge.Plot	1652
EWedgeGauge.PlotWithCurrentPen	1653
EWedgeGauge.Process	1654
EWedgeGauge.RectangularSamplingArea	1655
EWedgeGauge.RemoveAllSkipRanges	1655
EWedgeGauge.RemoveSkipRange	1655
EWedgeGauge.SamplingStep	1656
EWedgeGauge.SetDiameters	1656
EWedgeGauge.SetFromOriginMiddleEnd	1657
EWedgeGauge.SetFromTwoPoints	1657
EWedgeGauge.SetMinNumFitSamples	1658
EWedgeGauge.SetRadii	1659
EWedgeGauge.Shape	1659
EWedgeGauge.Smoothing	1660
EWedgeGauge.Thickness	1660
EWedgeGauge.Threshold	1660
EWedgeGauge.Tolerance	1661
EWedgeGauge.TransitionChoice	1661
EWedgeGauge.TransitionIndex	1662
EWedgeGauge.TransitionType	1662
EWedgeGauge.Type	1662
EWedgeGauge.Valid	1663
EWedgeGauge.Wedge	1663
EWedgeShape Class	1663
EWedgeShape.Amplitude	1668
EWedgeShape.Angle	1668
EWedgeShape.ApexAngle	1668
EWedgeShape.Breadth	1669

EWedgeShape.Center	1669
EWedgeShape.CenterX	1669
EWedgeShape.CenterY	1670
EWedgeShape.Closest	1670
EWedgeShape.CopyTo	1670
EWedgeShape.Direct	1671
EWedgeShape.Drag	1671
EWedgeShape.Draw	1672
EWedgeShape.DrawWithCurrentPen	1673
EWedgeShape.EndAngle	1673
EWedgeShape.EWedgeShape	1674
EWedgeShape.FullBreadth	1674
EWedgeShape.FullCircle	1674
EWedgeShape.GetCorners	1675
EWedgeShape.GetEdges	1675
EWedgeShape.GetInnerPoint	1676
EWedgeShape.GetMidEdges	1677
EWedgeShape.GetOuterPoint	1677
EWedgeShape.GetPoint	1678
EWedgeShape.HitTest	1678
EWedgeShape.InnerApex	1679
EWedgeShape.InnerArcLength	1679
EWedgeShape.InnerDiameter	1679
EWedgeShape.InnerEnd	1680
EWedgeShape.InnerOrg	1680
EWedgeShape.InnerRadius	1680
EWedgeShape.operator=	1681
EWedgeShape.OrgAngle	1681
EWedgeShape.OuterApex	1681
EWedgeShape.OuterArcLength	1682
EWedgeShape.OuterDiameter	1682
EWedgeShape.OuterEnd	1682
EWedgeShape.OuterOrg	1683
EWedgeShape.OuterRadius	1683
EWedgeShape.Scale	1683
EWedgeShape.SetCenterXY	1684
EWedgeShape.SetDiameters	1684
EWedgeShape.SetFromCenterAndOrigin	1685
EWedgeShape.SetFromOriginMiddleEnd	1685
EWedgeShape.SetFromTwoPoints	1686
EWedgeShape.SetRadii	1687
EWedgeShape.Type	1687
EWedgeShape.Wedge	1688
 EWorldShape Class	1688
EWorldShape.AddLandmark	1696
EWorldShape.AddPoint	1697
EWorldShape.Angle	1698
EWorldShape.AutoCalibrate	1699
EWorldShape.AutoCalibrateDotGrid	1699

EWorldShape.AutoCalibrateLandmarks	1700
EWorldShape.Calibrate	1701
EWorldShape.CalibrationModes	1701
EWorldShape.CalibrationSucceeded	1702
EWorldShape.Center	1702
EWorldShape.CenterX	1703
EWorldShape.CenterY	1703
EWorldShape.Closest	1703
EWorldShape.DisableTypeFilter	1704
EWorldShape.DistortionStrength	1704
EWorldShape.DistortionStrength2	1704
EWorldShape.Drag	1705
EWorldShape.DragLandmark	1705
EWorldShape.Draw	1706
EWorldShape.DrawCrossGrid	1706
EWorldShape.DrawCrossGridWithCurrentPen	1708
EWorldShape.DrawGrid	1709
EWorldShape.DrawGridWithCurrentPen	1709
EWorldShape.DrawLandmarks	1710
EWorldShape.DrawWithCurrentPen	1710
EWorldShape.EmptyLandmarks	1711
EWorldShape.EnableTypeFilter	1711
EWorldShape.EWorldShape	1712
EWorldShape.FieldHeight	1712
EWorldShape.FieldWidth	1713
EWorldShape.GetLandmarkElement	1713
EWorldShape.GridPointsMaxVariation	1714
EWorldShape.GridPointsMaxVariationThreshold	1714
EWorldShape.GridPointsMeanVariation	1715
EWorldShape.GridPointsMeanVariationThreshold	1715
EWorldShape.HitLandmark	1716
EWorldShape.HitLandmarks	1716
EWorldShape.HitTest	1717
EWorldShape.NumLandmarkElements	1717
EWorldShape.operator=	1717
EWorldShape.OpticalCenterX	1718
EWorldShape.OpticalCenterY	1718
EWorldShape.PanX	1718
EWorldShape.PanY	1719
EWorldShape.PerspectiveStrength	1719
EWorldShape.Ratio	1719
EWorldShape.RebuildGrid	1720
EWorldShape.RemoveLandmark	1721
EWorldShape.Scale	1721
EWorldShape.SensorHeight	1722
EWorldShape.SensorToWorld	1722
EWorldShape.SensorWidth	1722
EWorldShape.SetCenterXY	1723
EWorldShape.SetDistortion	1723

EWorldShape.SetFieldSize	1724
EWorldShape.SetPan	1725
EWorldShape.SetPerspective	1725
EWorldShape.SetResolution	1726
EWorldShape.SetSensor	1727
EWorldShape.SetSensorSize	1729
EWorldShape.GetSize	1729
EWorldShape.SetupUnwarp	1730
EWorldShape.SetZoom	1731
EWorldShape.TiltXAngle	1731
EWorldShape.TiltYAngle	1732
EWorldShape.Type	1732
EWorldShape.Unwarp	1733
EWorldShape.WorldToSensor	1734
EWorldShape.XResolution	1734
EWorldShape.YResolution	1734
EWorldShape.ZoomX	1735
EWorldShape.ZoomY	1735
 Structures	1736
E3DPoint Struct	1736
E3DPoint.E3DPoint	1736
E3DPoint.operator!=	1737
E3DPoint.operator==	1737
E3DPoint.X	1738
E3DPoint.Y	1738
E3DPoint.Z	1738
EBW1 Struct	1739
EBW1.EBW1	1739
EBW1.Size	1740
EBW1.UINT32Value	1740
EBW1.Value	1741
EBW16 Struct	1741
EBW16.EBW16	1742
EBW16.Size	1742
EBW16.UINT32Value	1742
EBW16.Value	1743
EBW16Path Struct	1743
EBW16Path.Pixel	1743
EBW16Path.X	1744
EBW16Path.Y	1744
EBW32 Struct	1744
EBW32.EBW32	1745
EBW32.Size	1745
EBW32.UINT32Value	1746
EBW32.Value	1746
EBW8 Struct	1746

EBW8.EBW8	1747
EBW8.Size	1747
EBW8.UINT32Value	1748
EBW8.Value	1748
EBW8Path Struct	1748
EBW8Path.Pixel	1749
EBW8Path.X	1749
EBW8Path.Y	1749
EC15 Struct	1750
EC15.C0	1750
EC15.C1	1751
EC15.C2	1751
EC15.EC15	1751
EC15.Size	1752
EC15.UINT32Value	1752
EC16 Struct	1753
EC16.C0	1753
EC16.C1	1754
EC16.C2	1754
EC16.EC16	1754
EC16.Size	1755
EC16.UINT32Value	1755
EC24 Struct	1756
EC24.C0	1756
EC24.C1	1757
EC24.C2	1757
EC24.EC24	1757
EC24.Size	1758
EC24.UINT32Value	1758
EC24A Struct	1759
EC24A.A	1760
EC24A.C0	1760
EC24A.C1	1760
EC24A.C2	1761
EC24A.EC24A	1761
EC24A.Size	1762
EC24A.UINT32Value	1762
EC24Path Struct	1762
EC24Path.Pixel	1763
EC24Path.X	1763
EC24Path.Y	1763
EC48 Struct	1763
EC48.BitsPerPixelCode	1764
EC48.C0	1765
EC48.C1	1765
EC48.C2	1765
EC48.DefaultColorSystem	1765

EC48.EC48	1766
EC48.IsEqual	1766
EC48.PlanesPerPixel	1767
EC48.Size	1767
EColor Struct	1768
EColor.C0	1768
EColor.C1	1768
EColor.C2	1769
EColor.EColor	1769
EDepth16 Struct	1770
EDepth16.EDepth16	1770
EDepth16.Size	1771
EDepth16(UINT32Value	1771
EDepth16.Value	1771
EDepth32f Struct	1772
EDepth32f.EDepth32f	1772
EDepth32f.FLOAT32Value	1773
EDepth32f.Size	1773
EDepth32f.Value	1773
EDepth8 Struct	1774
EDepth8.EDepth	1774
EDepth8.Size	1775
EDepth8(UINT32Value	1775
EDepth8.Value	1775
EFeatureData Struct	1776
EFeatureData.FeatDataSize	1776
EFeatureData.FeatDataType	1777
EFeatureData.FeatNum	1777
EFeatureData.Size	1777
EISH Struct	1777
EISH.H	1778
EISH.I	1778
EISH.S	1778
ELAB Struct	1779
ELAB.A	1779
ELAB.B	1779
ELAB.L	1780
ELCH Struct	1780
ELCH.C	1780
ELCH.H	1781
ELCH.L	1781
ELSH Struct	1781
ELSH.H	1782
ELSH.L	1782
ELSH.S	1782
ELUV Struct	1783

ELUV.L	1783
ELUV.U	1783
ELUV.V	1784
EMatchPosition Struct	1784
EMatchPosition.Angle	1785
EMatchPosition.AreaRatio	1785
EMatchPosition.CenterX	1786
EMatchPosition.CenterY	1786
EMatchPosition.Interpolated	1786
EMatchPosition.Scale	1787
EMatchPosition.ScaleX	1787
EMatchPosition.ScaleY	1787
EMatchPosition.Score	1788
EMatrixCodeIso15415GradingParameters Struct	1788
EMatrixCodeIso15415GradingParameters.AxialNonUniformity	1789
EMatrixCodeIso15415GradingParameters.AxialNonUniformityGrade	1790
EMatrixCodeIso15415GradingParameters.DecodingGrade	1790
EMatrixCodeIso15415GradingParameters.FixedPatternDamageGrade	1790
EMatrixCodeIso15415GradingParameters.GridNonUniformity	1791
EMatrixCodeIso15415GradingParameters.GridNonUniformityGrade	1791
EMatrixCodeIso15415GradingParameters.HorizontalPrintGrowth	1791
EMatrixCodeIso15415GradingParameters.ModulationGrade	1792
EMatrixCodeIso15415GradingParameters.OverallSymbolGrade	1792
EMatrixCodeIso15415GradingParameters.ReflectanceMarginGrade	1792
EMatrixCodeIso15415GradingParameters.SymbolContrast	1793
EMatrixCodeIso15415GradingParameters.SymbolContrastGrade	1793
EMatrixCodeIso15415GradingParameters.UnusedErrorCorrection	1793
EMatrixCodeIso15415GradingParameters.UnusedErrorCorrectionGrade	1794
EMatrixCodeIso15415GradingParameters.VerticalPrintGrowth	1794
EMatrixCodeIso29158GradingParameters Struct	1794
EMatrixCodeIso29158GradingParameters.CellContrastGrade	1795
EMatrixCodeIso29158GradingParameters.CellModulationGrade	1795
EMatrixCodeIso29158GradingParameters.FixedPatternDamageGrade	1796
EMatrixCodeIso29158GradingParameters.IsMeanLightInRequiredBounds	1796
EMatrixCodeIso29158GradingParameters.MeanLight	1796
EMatrixCodeIso29158GradingParameters.MinimumReflectanceGrade	1797
EMatrixCodeIso29158GradingParameters.OverallSymbolGrade	1797
EMatrixCodeSemiT10GradingParameters Struct	1797
EMatrixCodeSemiT10GradingParameters.CellDefects	1798
EMatrixCodeSemiT10GradingParameters.DataMatrixCellHeight	1799
EMatrixCodeSemiT10GradingParameters.DataMatrixCellWidth	1799
EMatrixCodeSemiT10GradingParameters.FinderPatternDefects	1799
EMatrixCodeSemiT10GradingParameters.HorizontalMarkGrowth	1800
EMatrixCodeSemiT10GradingParameters.HorizontalMarkMisplacement	1800
EMatrixCodeSemiT10GradingParameters.SymbolContrast	1800
EMatrixCodeSemiT10GradingParameters.SymbolContrastSNR	1801
EMatrixCodeSemiT10GradingParameters.UnusedErrorCorrection	1801
EMatrixCodeSemiT10GradingParameters.VerticalMarkGrowth	1801

EMatrixCodeSemiT10GradingParameters.VerticalMarkMisplacement	1802
EObjectData Struct	1802
EObjectData.Class	1803
EObjectData.IsHole	1803
EObjectData.isSelected	1804
EObjectData.ObjNbHole	1804
EObjectData.ObjNbRun	1804
EObjectData.ObjNum	1804
EOCR2CharacterCandidate Struct	1805
EOCR2CharacterCandidate.Code	1805
EOCR2CharacterCandidate.EOCR2CharacterCandidate	1806
EOCR2CharacterCandidate.Score	1806
EPath Struct	1807
EPath.X	1807
EPath.Y	1807
EPeak Struct	1808
EPeak.Amplitude	1808
EPeak.Area	1808
EPeak.Center	1809
EPeak.Length	1809
EPeak.Start	1809
ERGB Struct	1810
ERGB.B	1810
ERGB.G	1810
ERGB.R	1811
ERGBCColor Struct	1811
ERGBCColor.Blue	1811
ERGBCColor.ERGBCColor	1812
ERGBCColor.Green	1812
ERGBCColor.Red	1813
ERunData Struct	1813
ERunData.Class	1814
ERunData.Len	1814
ERunData.ObjNum	1814
ERunData.OrgX	1814
ERunData.OrgY	1815
ETransitionData Struct	1815
ETransitionData.Contrast	1816
ETransitionData.Location	1816
ETransitionData.MaxSlope	1817
ETransitionData.Polarity	1817
ETransitionData.Score	1817
ETransitionData.Width	1817
EVSH Struct	1818
EVSH.H	1818
EVSH.S	1818

EVSH.V	1819
EXYZ Struct	1819
EXYZ.X	1819
EXYZ.Y	1820
EXYZ.Z	1820
EYIQ Struct	1820
EYIQ.I	1821
EYIQ.Q	1821
EYIQ.Y	1821
EYSH Struct	1822
EYSH.H	1822
EYSH.S	1822
EYSH.Y	1823
EYUV Struct	1823
EYUV.U	1823
EYUV.V	1824
EYUV.Y	1824
 Enumerations	1825
EAdaptiveThresholdMethod Enum	1825
EAngleUnit Enum	1825
EARithmeticLogicOperation Enum	1826
EasyOCR2CharacterFilter Enum	1829
EasyOCR2CharSpacingBias Enum	1829
EasyOCR2CharWidthBias Enum	1830
EasyOCR2DrawDetectionStyle Enum	1830
EasyOCR2DrawRecognitionStyle Enum	1831
EasyOCR2DrawSegmentationStyle Enum	1832
EasyOCR2TextPolarity Enum	1833
ECalibrationMode Enum	1833
ECannyThresholdingMode Enum	1834
ECharCreationMode Enum	1834
EClippingMode Enum	1835
EColorQuantization Enum	1835
EColorSystem Enum	1836
EConnexity Enum	1837
EContourMode Enum	1837
EContourThreshold Enum	1838
ECorrelationMode Enum	1839
EDataSize Enum	1839
EDataType Enum	1840

EDegreesOfFreedom Enum	1840
EDiagnostic Enum	1841
EDoubleThresholdMode Enum	1842
EDraggingMode Enum	1842
EDragHandle Enum	1842
EDrawableFeature Enum	1845
EDrawingMode Enum	1846
EEncodingConnexity Enum	1847
EError Enum	1847
EFamily Enum	1874
EFeature Enum	1875
EFilteringMode Enum	1880
EFindContrastMode Enum	1881
EFlipping Enum	1881
EFramePosition Enum	1882
EGayscaleSingleThreshold Enum	1882
EHarrisThresholdingMode Enum	1883
EHistogramFeature Enum	1884
EHitAndMissValue Enum	1885
EImageFileType Enum	1885
EImageType Enum	1886
EKernelRectifier Enum	1887
EKernelRotation Enum	1887
EKernelType Enum	1888
ELearningMode Enum	1890
ELearnParam Enum	1890
ELegacyFeature Enum	1891
ELocalSearchMode Enum	1896
ELocationMode Enum	1897
ELogicalSize Enum	1897
EMatchContrastMode Enum	1901
EMatchingMode Enum	1901
EMatrixCodeContrastMode Enum	1902
EMaximumAnalysisMode Enum	1902
ENormalizationMode Enum	1902
EOCRClass Enum	1903
EOCRCColor Enum	1905
EPatternType Enum	1906

EPickingMode Enum	1906
EPlotItem Enum	1907
EQRCodeCodingMode Enum	1907
EQRCodeEncoding Enum	1908
EQRCodeLevel Enum	1908
EQRCodeModel Enum	1909
EQRCodePerspectiveMode Enum	1909
EQRCodeScanPrecision Enum	1910
EQRDetectionMethod Enum	1910
EQRDetectionTradeOff Enum	1911
EQualityIndicator Enum	1912
ERectangleMode Enum	1912
EReductionMode Enum	1913
EReferenceNoise Enum	1914
ERgbStandard Enum	1914
ERoiHit Enum	1915
ESegmentationMethod Enum	1915
ESegmentationMode Enum	1916
ESelectByPosition Enum	1917
ESelectionFlag Enum	1918
ESelectOption Enum	1918
ESerializer.FileWriterMode Enum	1919
EShapeBehavior Enum	1920
EShapeType Enum	1921
EShiftingMode Enum	1921
ESingleThresholdMode Enum	1922
ESortDirection Enum	1922
ESortOption Enum	1923
EStockMeasurementUnit Enum	1923
ESymbologies Enum	1924
EThinStructureMode Enum	1927
EThresholdMode Enum	1927
ETransitionChoice Enum	1928
ETransitionType Enum	1928
Features Enum	1929

Libraries

Easy3D Library

CLASSES

[EDepthMap8](#)
[EDepthMap16](#)
[EDepthMap32f](#)
[E3DPointCloud](#)
[ELaserLineExtractor](#)
[E3DDepthMapToPointCloud](#)
[E3DExplicitGeometricCalibration](#)
[E3DPointCloudViewer](#)
[E3DSimpleCropper](#)
[E3DRectangularCropper](#)
[E3DSphericalCropper](#)
[E3DAffineTransformer](#)
[E3DAffineTransformMatrix](#)
[E3DPointCloudFactory](#)

STRUCTS

[E3DPoint](#)
[EDepth8](#)
[EDepth16](#)
[EDepth32f](#)

ENUMERATIONS

[EMaximumAnalysisMode](#)

EasylImage Library

CLASSES

[EasylImage](#)
[EKernel](#)
[EMovingAverage](#)

ENUMERATIONS

[EArithmeticLogicOperation](#)
[EContourMode](#)
[EContourThreshold](#)
[EHistogramFeature](#)
[EKernelRectifier](#)
[EKernelRotation](#)
[EKernelType](#)
[EReferenceNoise](#)
[EThresholdMode](#)

EasyColor Library

CLASSES

[EasyColor](#)
[EColorLookup](#)
[EPseudoColorLookup](#)

ENUMERATIONS

[EColorQuantization](#)
[EColorSystem](#)
[ERgbStandard](#)

EasyObject Library

CLASSES

[EasyObject](#)
[ECodedImage2](#)
[ECodedElement](#)
[EObject](#)
[EHole](#)
[EObjectSelection](#)
[EImageEncoder](#)
[EImageSegmenter](#)
[ETwoLayersImageSegmenter](#)
[EThreeLayersImageSegmenter](#)
[EBinaryImageSegmenter](#)
[EGayscaleSingleThresholdSegmenter](#)
[EGayscaleDoubleThresholdSegmenter](#)
[EColorSingleThresholdSegmenter](#)
[EColorRangeThresholdSegmenter](#)
[EImageRangeSegmenter](#)
[EReferencelImageSegmenter](#)
[ELabeledImageSegmenter](#)
[EObjectRunsIterator](#)

ENUMERATIONS

[EEncodingConnexity](#)
[ESegmentationMethod](#)
[ESingleThresholdMode](#)
[EDoubleThresholdMode](#)
[EFeature](#)

EasyMatch Library

CLASSES

[EMatcher](#)

STRUCTS

[EMatchPosition](#)

ENUMERATIONS

[ECorrelationMode](#)

[EMatchContrastMode](#)

[EFilteringMode](#)

EasyFind Library

CLASSES

[EFoundPattern](#)

[EPatternFinder](#)

ENUMERATIONS

[EFindContrastMode](#)

[ELocalSearchMode](#)

[EPatternType](#)

[EReductionMode](#)

[EThinStructureMode](#)

EasyGauge Library

CLASSES

[ECircleGauge](#)
[ELineGauge](#)
[EPointGauge](#)
[ERectangleGauge](#)
[EWedgeGauge](#)
[EFrameShape](#)

ENUMERATIONS

[EClippingMode](#)
[EPlotItem](#)
[ETransitionChoice](#)
[ETransitionType](#)

EasyOCR Library

CLASSES

[EOCR](#)

ENUMERATIONS

[EMatchingMode](#)
[EShiftingMode](#)
[EOCRClass](#)
[EOCRCColor](#)
[ESegmentationMode](#)

EasyOCV Library

CLASSES

[EOCV](#)
[EOCVChar](#)
[EOCVText](#)
[EChecker](#)

ENUMERATIONS

[ECharCreationMode](#)
[EDegreesOfFreedom](#)
[EDiagnostic](#)
[ELearningMode](#)
[ELocationMode](#)
[ENormalizationMode](#)
[EQualityIndicator](#)

EasyBarCode Library

CLASSES

[EBarcode](#)

ENUMERATIONS

[ESymbologies](#)

EasyMatrixCode Library

CLASSES

[EMatrixCode](#)
[EMatrixCodeReader](#)
[ESearchParamsType](#)

ENUMERATIONS

[EFamily](#)
[EFlipping](#)
[ELearnParam](#)
[ELogicalSize](#)
[EMatrixCodeContrastMode](#)

EasyQRCode Library

CLASSES

[EQRCode](#)
[EQRCodeDecodedStream](#)
[EQRCodeDecodedStreamPart](#)
[EQRCodeGeometry](#)
[EQRCodeReader](#)
[EQuadrilateral](#)

ENUMERATIONS

[EQRCodeCodingMode](#)
[EQRCodeEncoding](#)
[EQRCodeLevel](#)
[EQRCodeModel](#)
[EQRCodeScanPrecision](#)

Legacy

EasyObject Library (Legacy)

CLASSES

[ECodedImage](#)

ENUMERATIONS

[EConnexity](#)

[ELegacyFeature](#)

[ESelectByPosition](#)

[ESelectOption](#)

[ESortOption](#)

FUNCTIONS

[EasyObject::ContourArea](#)

[EasyObject::ContourGravityCenter](#)

[EasyObject::ContourInertia](#)

Classes

E3DAffineTransformer Class

Manages a 3D coordinates transformation context.

Remarks

By default, no transformation is done (identity matrix). The transformations are applied in the order in which the calls to AddTransform are done.

PROPERTIES

Transform

M

Transformation matrix.

E

THODS

AddAnisotropicScalingTransform

Adds anisotropic scaling to the current transformation matrix.

AddIsotropicScalingTransform

Adds isotropic scaling to the current transformation matrix.

AddOrthographicProjectionTransfo rm

Adds an orthographic projection to the current transformation matrix.

AddPerspectiveProjectionTransform	Adds a perspective projection to the current transformation matrix.
AddRotationXTransform	Adds rotation around the X axis to the current transformation matrix.
AddRotationYTransform	Adds rotation around the Y axis to the current transformation matrix.
AddRotationZTransform	Adds rotation around the Z axis to the current transformation matrix.
AddTransform	Compose a custom transformation with the current transformation matrix.
AddTranslationTransform	Adds translation to the current transformation matrix.
ApplyMatrix	Apply a transformation matrix to a point cloud.
ApplyTransform	Apply current transformation matrix to a point cloud.
CreateAnisotropicScalingMatrix	Creates an anisotropic scaling matrix.
CreateIdentityMatrix	Creates an identity (neutral) matrix.

[CreateIsotropicScalingMatrix](#) Creates an isotropic scaling matrix.

[CreateOrthographicProjectionMatrix](#) Creates an orthographic projection matrix.

[CreatePerspectiveProjectionMatrix](#)

Creates a perspective projection matrix.

CreateRotationXMatrix	Creates a rotation around the X axis matrix.
CreateRotationYMatrix	Creates a rotation around the Y axis matrix.
CreateRotationZMatrix	Creates a rotation around the Z axis matrix.
CreateTranslationMatrix	Creates a translation matrix.
E3DAffineTransform	Creates an E3DAffineTransform object.
Reset	Resets the transformation matrix to the identity.

3

DAffineTransform.AddAnisotropicScalingTransform

Adds anisotropic scaling to the current transformation matrix.

[VB6]

```
E3DAffineTransform AddAnisotropicScalingTransform(
    Single scaleX,
    Single scaleY,
    Single scaleZ
)
```

Parameters

scaleX

Scaling factor along the X axis.

scaleY

Scaling factor along the Y axis.

scaleZ

Scaling factor along the Z axis.

E3DAffineTransform.AddIsotropicScalingTransform

Adds isotropic scaling to the current transformation matrix.

[VB6]

```
E3DAffineTransformMatrix AddIsotropicScalingTransform(  
    Single scale  
)
```

Parameters

scale

Scaling factor.

E3DAffineTransform.AddOrthographicProjectionT ransform

Adds an orthographic projection to the current transformation matrix.

[VB6]

```
E3DAffineTransformMatrix AddOrthographicProjectionTransform(  
    Single width,  
    Single height  
)
```

Parameters

width

Width of the viewport.

height

Height of the viewport.

E3DAffineTransform.AddPerspectiveProjectionTransform

Adds a perspective projection to the current transformation matrix.

[VB6]

```
E3DAffineTransformMatrix AddPerspectiveProjectionTransform(  
    Single distance,  
    Single width,  
    Single height  
)
```

Parameters

distance

Distance of the viewport to the origin.

width

Width of the viewport.

height

Height of the viewport.

E3DAffineTransform.AddRotationXTransform

Adds rotation around the X axis to the current transformation matrix.

[VB6]

```
E3DAffineTransformMatrix AddRotationXTransform(  
    Single Angle  
)
```

Parameters

Angle

Rotation angle.

E3DAffineTransformer.AddRotationYTransform

Adds rotation around the Y axis to the current transformation matrix.

[VB6]

```
E3DAffineTransformMatrix AddRotationYTransform(  
    Single Angle  
)
```

Parameters

Angle

Rotation angle.

E3DAffineTransformer.AddRotationZTransform

Adds rotation around the Z axis to the current transformation matrix.

[VB6]

```
E3DAffineTransformMatrix AddRotationZTransform(  
    Single Angle  
)
```

Parameters

Angle

Rotation angle.

E3DAffineTransform.AddTransform

Compose a custom transformation with the current transformation matrix.

[VB6]

```
E3DAffineTransformMatrix AddTransform(  
    E3DAffineTransformMatrix matrix  
)
```

Parameters

matrix

Transformation matrix.

E3DAffineTransform.AddTranslationTransform

Adds translation to the current transformation matrix.

[VB6]

```
E3DAffineTransformMatrix AddTranslationTransform(  
    Single dX,  
    Single dY,  
    Single dZ  
)
```

Parameters

dX

Translation along the X axis.

dY

Translation along the Y axis.

dZ

Translation along the Z axis.

E3DAffineTransformer.ApplyMatrix

Apply a transformation matrix to a point cloud.

[VB6]

```
void ApplyMatrix(  
    E3DAffineTransformMatrix matrix,  
    E3DPointCloud cloud,  
    E3DPointCloud transformedCloud  
)
```

Parameters

matrix

-

cloud

Transformation matrix.

transformedCloud

Transformed cloud.

E3DAffineTransformer.ApplyTransform

Apply current transformation matrix to a point cloud.

[VB6]

```
void ApplyTransform(  
    E3DPointCloud cloud,  
    E3DPointCloud transformedCloud  
)
```

Parameters

cloud

Cloud to transform.

transformedCloud

Transformed cloud.

E3DAffineTransformer.CreateAnisotropicScalingMatrix

Creates an anisotropic scaling matrix.

[VB6]

```
E3DAffineTransformMatrix CreateAnisotropicScalingMatrix(  
    Single scaleX,  
    Single scaleY,  
    Single scaleZ  
)
```

Parameters

scaleX

Scaling factor along the X axis.

scaleY

Scaling factor along the Y axis.

scaleZ

-

E3DAffineTransformer.CreateIdentityMatrix

Creates an identity (neutral) matrix.

[VB6]

```
E3DAffineTransformMatrix CreateIdentityMatrix(
)
```

E3DAffineTransformer.CreateIsotropicScalingMatrix

Creates an isotropic scaling matrix.

[VB6]

```
E3DAffineTransformMatrix CreateIsotropicScalingMatrix(
    Single scale
)
```

Parameters

scale

Scaling factor.

E3DAffineTransformer.CreateOrthographicProjectio nMatrix

Creates an orthographic projection matrix.

[VB6]

```
E3DAffineTransformMatrix CreateOrthographicProjectionMatrix(
    Single width,
    Single height
)
```

Parameters

width

Width of the viewport.

height

Height of the viewport.

E3DAffineTransform.CreatePerspectiveProjectionMatrix

Creates a perspective projection matrix.

[VB6]

```
E3DAffineTransform CreatePerspectiveProjectionMatrix(  
    Single distance,  
    Single width,  
    Single height  
)
```

Parameters

distance

Distance of the viewport to the origin.

width

Width of the viewport.

height

Height of the viewport.

E3DAffineTransform.CreateRotationXMatrix

Creates a rotation around the X axis matrix.

[VB6]

```
E3DAffineTransform CreateRotationXMatrix(  
    Single Angle  
)
```

Parameters

Angle

Rotation angle.

E3DAffineTransformer.CreateRotationYMatrix

Creates a rotation around the Y axis matrix.

[VB6]

```
E3DAffineTransformMatrix CreateRotationYMatrix(
    Single Angle
)
```

Parameters

Angle

Rotation angle.

E3DAffineTransformer.CreateRotationZMatrix

Creates a rotation around the Z axis matrix.

[VB6]

```
E3DAffineTransformMatrix CreateRotationZMatrix(
    Single Angle
)
```

Parameters

Angle

Rotation angle.

E3DAffineTransforme.CreateTranslationMatrix

Creates a translation matrix.

[VB6]

```
E3DAffineTransformMatrix CreateTranslationMatrix(  
    Single dX,  
    Single dY,  
    Single dZ  
)
```

Parameters

dX

Translation along the X axis.

dY

Translation along the Y axis.

dZ

Translation along the Z axis.

E3DAffineTransforme.E3DAffineTransforme

Creates an [E3DAffineTransforme](#) object.

[VB6]

```
void E3DAffineTransforme(  
)
```

E3DAffineTransforme.Reset

Resets the transformation matrix to the identity.

[VB6]

```
void Reset()  
)
```

E3DAffineTransforme.Transform

Transformation matrix.

[VB6]

```
Transform As E3DAffineTransformMatrix  
read-write
```

E3DAffineTransformMatrix Class

Represents a 3D affine transformation [4x4] matrix .

METHODS

E3DAffineTransformMatrix

Creates an [E3DAffineTransformMatrix](#) object.

GetValue

Gets a value from the matrix.

operator*

Matrix product. Combines the transformations of the two matrices.

SetValue

Sets a value in the matrix.

E3DAffineTransformMatrix.E3DAffineTransformMatrix

X

Creates an [E3DAffineTransformMatrix](#) object.

```
[VB6]  
void E3DAffineTransformMatrix()  
)
```

E3DAffineTransformMatrix.GetValue

Gets a value from the matrix.

```
[VB6]  
Single GetValue(  
Long column,  
Long row  
)
```

Parameters

column

Column of the value to set.

row

Row of the value to set.

E3DAffineTransformMatrix.operator*

Matrix product. Combines the transformations of the two matrices.

[VB6]

```
E3DAffineTransformMatrix operator*(
    E3DAffineTransformMatrix M2
)
```

Parameters

M2

-

E3DAffineTransformMatrix.SetValue

Sets a value in the matrix.

[VB6]

```
void SetValue(
    Long column,
    Long row,
    Single value
)
```

Parameters

column

Column of the value to set.

row

Row of the value to set.

value

Value to set.

E3DDepthMapToPointCloud Class

Manages a simple conversion of Depth Map to Point Cloud. A scale factor is used to transform (u, v, depth) coordinate to (x, y, z) point

METHODS

Apply

Applies the dummy calibration to the given depthMap and stores the transformed points in the given point cloud.

E3DDepthMapToPointCloud

Constructs a E3DDepthMapToPointCloud. E3DDepthMapToPointCloud is used to convert a depth map to a point cloud, given some pixel to mm scale values. That's not really a calibration (perspective is not corrected) but a simple way to produce a point cloud given some scale factors.

DDepthMapToPointCloud.Apply

Applies the dummy calibration to the given depthMap and stores the transformed points in the given point cloud.

[VB6]

```
void Apply(
    EDepthMap8 srcDepthMap,
    E3DPointCloud dstPointCloud
)

void Apply(
    EDepthMap16 srcDepthMap,
    E3DPointCloud dstPointCloud
)
```

```
void Apply(
    EDepthMap32f srcDepthMap,
    E3DPointCloud dstPointCloud
)
```

Parameters*srcDepthMap*

The source depth map.

dstPointCloud

The destination point cloud.

E3DDepthMapToPointCloud.E3DDepthMapToPointCloud

Constructs a [E3DDepthMapToPointCloud](#). E3DDepthMapToPointCloud is used to convert a depth map to a point cloud, given some pixel to mm scale values. That's not really a calibration (perspective is not corrected) but a simple way to produce a point cloud given some scale factors.

[VB6]

```
void E3DDepthMapToPointCloud(
    Single factorX,
    Single factorY,
    Single factorZ
)
```

Parameters*factorX*

Width of pixel in mm (factor for X coordinate).

factorY

Distance between 2 profiles (depth map lines) in mm (factor for Y coordinate).

factorZ

Scale of the pixel value in mm (factor Z coordinate).

E3DExplicitGeometricCalibration Class

Manages a 3D explicit geometric calibration context.

PROPERTIES

CameraAngle

-

CameraPlaneDistance

-

FocalLength

-

MotionIncrement

-

PlaneAngle

-

SensorHeight

-

SensorWidth

M
E

THODS

Apply

Apply the explicit geometric calibration to the given depth map and store the transformed points in the given point cloud.

[E3DExplicitGeometricCalibration](#)

Constructs a [E3DExplicitGeometricCalibration](#).
 E3DExplicitGeometricCalibration is used to
 calibrate a depth map from a minimal set of
 explicit geometric values.

DExplicitGeometricC alibration.Apply

Apply the explicit geometric calibration to the given depth map and store the transformed points in the given point cloud.

[VB6]

```
void Apply(
  EDepthMap8 srcDepthMap,
  Long roiOrigin,
  Long roiHeight,
  E3DPointCloud dstPointCloud
)

void Apply(
  EDepthMap16 srcDepthMap,
  Long roiOrigin,
  Long roiHeight,
  E3DPointCloud dstPointCloud
)

void Apply(
  EDepthMap32f srcDepthMap,
  Long roiOrigin,
  Long roiHeight,
  E3DPointCloud dstPointCloud
)
```

Parameters

srcDepthMap

The source depth map.

roiOrigin

The ROI origin line (in original source image) used in laser line extraction. Between the top (0) and the bottom (height) of the image.

roiHeight

-
dstPointCloud

The destination point cloud. The coordinate's origin is the camera focal point, X is aligned with the sensor width, Y is horizontal (aligned to the motion), Z is facing down. Values are expressed in mm.

E3DExplicitGeometricCalibration.CameraAngle

-

[VB6]

CameraAngle As Single

read-only

E3DExplicitGeometricCalibration.CameraPlaneDist ance

-

[VB6]

CameraPlaneDistance As Single

read-only

E3DExplicitGeometricCalibration.E3DExplicitGeometricCalibration

Constructs a [E3DExplicitGeometricCalibration](#). E3DExplicitGeometricCalibration is used to calibrate a depth map from a minimal set of explicit geometric values.

[VB6]

```
void E3DExplicitGeometricCalibration(
    Single sensorWidth,
    Single sensorHeight,
    Single focalLength,
    Single cameraAngle,
    Single planeAngle,
    Single cameraPlaneDistance,
    Single motionIncrement
)
```

Parameters

sensorWidth

The camera sensor width, in mm.

sensorHeight

The camera sensor height, in mm.

focalLength

The camera optics focal length, in mm.

cameraAngle

The camera angle from the vertical axis. Looking down camera is angle 0. Angle is expressed in radians, range from 0 to PI/2 and is positive in counter clockwise direction.

planeAngle

The laser plane angle from the vertical axis. A perfect vertical laser orientation is angle 0. Angle is expressed in radians, range from -PI/2 to PI/2 and is positive in counter clockwise direction.

cameraPlaneDistance

The horizontal distance in mm from the optics focal point to the laser plane.

motionIncrement

The distance in mm between each line of the depth map. That's the relative motion of the camera/laser setup to the object position.

E3DExplicitGeometricCalibration.FocalLength

-

[VB6]

FocalLength As Single

read-only

E3DExplicitGeometricCalibration.MotionIncrement

-

[VB6]

MotionIncrement As Single

read-only

E3DExplicitGeometricCalibration.PlaneAngle

-

[VB6]

PlaneAngle As Single

read-only

E3DExplicitGeometricCalibration.SensorHeight

-

[VB6]

SensorHeight As Single

read-only

E3DExplicitGeometricCalibration.SensorWidth

-

[VB6]

SensorWidth As Single

read-only

E3DPointCloud Class

Represents a 3D point cloud.

PROPERTIES

NumPoints

Number of points in the point cloud.

Points

Retrieves the points of the point cloud in the form of a vector.

PointsBuffer	M Retrieves a pointer to the internal points buffer.
THODS	E
AddPoint	Adds a point to the point cloud.
AddPointCloud	Adds the points of another point cloud to the point cloud.
AddPoints	Adds a vector of points to the point cloud.
Clear	Empties the point cloud.
E3DPointCloud	Creates an @E3DSpherePointCloud@ object.
FillPointsBuffer	Copies an external points buffer into the internal points buffer.
GetPoint	Retrieves a point from the point cloud.
LoadPCD	-
SavePCD	-

Serialize

|E-

3

DPointCloud.AddPoint

Adds a point to the point cloud.

[VB6]

```
void AddPoint(  
    E3DPoint point  
)
```

Parameters

point

Point to add to the cloud.

E3DPointCloud.AddPointCloud

Adds the points of another point cloud to the point cloud.

[VB6]

```
void AddPointCloud(  
    E3DPointCloud cloud  
)
```

Parameters

cloud

Cloud whose points will be added to the cloud.

E3DPointCloud.AddPoints

Adds a vector of points to the point cloud.

```
[VB6]  
void AddPoints(  
    ()E3DPoint points  
)
```

Parameters

points
Vector of points to add to the cloud.

E3DPointCloud.Clear

Empties the point cloud.

```
[VB6]  
void Clear(  
)
```

E3DPointCloud.E3DPointCloud

Creates an @E3DSpherePointCloud@ object.

```
[VB6]  
void E3DPointCloud(  
)
```

E3DPointCloud.FillPointsBuffer

Copies an external points buffer into the internal points buffer.

```
[VB6]  
void FillPointsBuffer(  
    Long pointsBuffer,  
    Long numPoints  
)
```

Parameters

pointsBuffer

Address of the extrnal points buffer.

numPoints

Number of points in the external points buffer.

Remarks

The buffer must contain points in the form of triplets of 32bits floats stored in the (X,Y,Z) order.

E3DPointCloud.GetPoint

Retrieves a point from the point cloud.

```
[VB6]  
E3DPoint GetPoint(  
    Long index  
)
```

Parameters

index

Index of the point to be retrieved.

E3DPointCloud.LoadPCD

-

[VB6]

```
void LoadPCD(  
    String path  
)
```

Parameters

path

E3DPointCloud.NumPoints

Number of points in the point cloud.

[VB6]

```
NumPoints As Long  
read-only
```

E3DPointCloud.Points

Retrieves the points of the point cloud in the form of a vector.

[VB6]

```
Points As ()E3DPoint
```

read-only

E3DPointCloud.PointsBuffer

Retrieves a pointer to the internal points buffer.

[VB6]

PointsBuffer As Long

read-only

E3DPointCloud.SavePCD

-

[VB6]

```
void SavePCD(  
    String path  
)
```

Parameters

path

-

E3DPointCloud.Serialize

-

[VB6]

```
void Serialize(  
    ESerializer serializer  
)
```

Parameters

serializer

-

E3DPointCloudFactory Class

Manages a context for creating point clouds of specific shapes.

METHODS

CreateCubicPointCloud

Creates a point cloud in the shape of a cube.

CreateRectangularPointCloud

Creates a point cloud in the shape of a rectangular parallelepiped.

CreateSphericPointCloud

Creates a point cloud in the shape of a sphere.

3

DPointCloudFactory.CreateCubicPointCloud

Creates a point cloud in the shape of a cube.

[VB6]

```
E3DPointCloud CreateCubicPointCloud(  
    E3DPoint center,  
    Single size,  
    Single roll,  
    Single pitch,  
    Single yaw,  
    Long numSamples  
)
```

Parameters

center

Center of the cube.

size

Edge size of the cube.

roll

Roll (rotation along the X axis) of the cube.

pitch

Pitch (rotation along the Y axis) of the cube.

yaw

Yaw (rotation along the Z axis) of the cube.

numSamples

Number of points along each edge of the cube.

E3DPointCloudFactory.CreateRectangularPointCloud

Creates a point cloud in the shape of a rectangular parallelepiped.

[VB6]

```
E3DPointCloud CreateRectangularPointCloud(  
    E3DPoint center,  
    Single width,  
    Single height,  
    Single depth,  
    Single roll,  
    Single pitch,  
    Single yaw,  
    Long numSamples  
)
```

Parameters

center

Center of the rectangular parallelepiped.

width

Width (size along the X axis before rotation) of the rectangular parallelepiped.

height

Height (size along the Y axis before rotation) of the rectangular parallelepiped.

depth

Depth (size along the Z axis before rotation) of the rectangular parallelepiped.

roll

Roll (rotation along the X axis) of the rectangular parallelepiped.

pitch

Pitch (rotation along the Y axis) of the rectangular parallelepiped.

yaw

Yaw (rotation along the Z axis) of the rectangular parallelepiped.

numSamples

Number of points along each edge of the rectangular parallelepiped.

E3DPointCloudFactory.CreateSphericPointCloud

Creates a point cloud in the shape of a sphere.

[VB6]

```
E3DPointCloud CreateSphericPointCloud(  
    E3DPoint center,  
    Single radius,  
    Long numCircles,  
    Long numSamples  
)
```

Parameters

center

Center of the sphere.

radius

Radius of the sphere.

numCircles

Number of parallels and meridians to be rendered.

numSamples

Number of points along each meridian and parallel.

E3DPointCloudStatistics Class

Manages a context for retrieving statistics on a point cloud.

METHODS

[GetPointCloudBounds](#)

Retrieves the bounds of a point cloud.

[GetPointCloudCenterOfGravity](#)

Retrieves the center of gravity of a point cloud.

3

DPointCloudStatistics.GetPointCloudBounds

Retrieves the bounds of a point cloud.

[VB6]

```
void GetPointCloudBounds(  
    E3DPointCloud cloud,  
    EFloatRange rangeX,  
    EFloatRange rangeY,  
    EFloatRange rangeZ  
)
```

Parameters

cloud

Point cloud.

rangeX

Bounds of the point cloud along the X direction.

rangeY

Bounds of the point cloud along the Y direction.

rangeZ

Bounds of the point cloud along the Z direction.

E3DPointCloudStatistics.GetPointCloudCenterOfGravity

Retrieves the center of gravity of a point cloud.

[VB6]

```
E3DPoint GetPointCloudCenterOfGravity(  
    E3DPointCloud cloud  
)
```

Parameters

cloud

Point cloud.

E3DPointCloudViewer Class

Manages a viewer window for point clouds.

METHODS

[ConfigureViewport](#)

Configures the point cloud to be rendered.

[E3DPointCloudViewer](#)

Creates an [E3DPointCloudViewer](#) object.

[SetCloud](#)

Sets the point cloud to be rendered.

[SetPosition](#)

Sets the position of the viewer window.

[SetViewingAngle](#)

Set viewing angle.

[Show](#)

Shows the viewer window.

3

DPointCloudViewer.ConfigureViewport

Configures the point cloud to be rendered.

[VB6]

```
void ConfigureViewport(  
    Single viewportSize,  
    Single pivotX,  
    Single pivotY  
)
```

Parameters

viewportSize

Size of the viewing volume.

pivotX

X offset of the rotation pivot point.

pivotY

Y offset of the rotation pivot point.

E3DPointCloudViewer.E3DPointCloudViewer

Creates an [E3DPointCloudViewer](#) object.

[VB6]

```
void E3DPointCloudViewer(  
    Long orgX,  
    Long orgY,  
    Long width,  
    Long height,  
    Long parent  
)
```

Parameters

orgX

X coordinate of the top left corner of the viewer window.

orgY

Y coordinate of the top left corner of the viewer window.

width

Width of the viewer window.

height

height of the viewer window.

parent

Handle (HWND) of the parent window of the viewer. If NULL, the viewer is built as a independent floating window.

E3DPointCloudViewer.SetCloud

Sets the point cloud to be rendered.

[VB6]

```
void SetCloud(  
    E3DPointCloud cloud,  
    Single viewportSize,  
    Single pivotX,  
    Single pivotY  
)
```

Parameters

cloud

Cloud to render.

viewportSize

Size of the viewing volume.

pivotX

X offset of the rotation pivot point.

pivotY

Y offset of the rotation pivot point.

Remarks

For display performance purposes, the point cloud is copied into the viewer. Subsequent modifications on the point cloud thus not be visible until a new call to [SetCloud](#) has been made.

E3DPointCloudViewer.SetPosition

Sets the position of the viewer window.

[VB6]

```
void SetPosition(  
    Long orgX,  
    Long orgY,  
    Long width,  
    Long height  
)
```

Parameters

orgX

-

orgY

-

width

-

height

-

E3DPointCloudViewer.SetViewingAngle

Set viewing angle.

[VB6]

```
void SetViewingAngle(  
    Single angleX,  
    Single angleY  
)
```

Parameters

angleX

Rotation around the X axis.

angleY

Rotation around the Y axis..

E3DPointCloudViewer.Show

Shows the viewer window.

```
[VB6]  
void Show()  
)
```

E3DRectangularCropper Class

Manages a point cloud cropper in the shape of a rectangular parallelepiped.

METHODS

Crop

Crops a point cloud.

E3DRectangularCropper

Creates an [E3DRectangularCropper](#) object.

3

DRectangularCropper.Crop

Crops a point cloud.

```
[VB6]
```

```
void Crop(
    E3DPointCloud cloudIn,
    E3DPointCloud cloudOut,
    Boolean invert
)
```

Parameters*cloudIn*

Cloud to be cropped.

cloudOut

Cropped cloud.

invert

E3DRectangularCropper.E3DRectangularCropper

Creates an [E3DRectangularCropper](#) object.

[VB6]

```
void E3DRectangularCropper(
    E3DPoint center,
    Single width,
    Single height,
    Single depth,
    Single roll,
    Single pitch,
    Single yaw
)
```

Parameters*center*

Center of the rectangular parallelepiped.

width

Width of the rectangular parallelepiped.

height

Height of the rectangular parallelepiped.

depth

Depth of the rectangular parallelepiped.

roll

Roll (rotation along the X axis) of the rectangular parallelepiped.

pitch

Pitch (rotation along the Y axis) of the rectangular parallelepiped.

yaw

Yaw (rotation along the Z axis) of the rectangular parallelepiped.

E3DSimpleCropper Class

Manages a point cloud cropper based on X/Y/Z value ranges.

PROPERTIES

XRange

Sets the allowed X value range.

YRange

Sets the allowed Y value range.

ZRange

M Sets the allowed Z value range.

E

THODS

Crop

Crops a point cloud.

E3DSimpleCropper

Creates an [E3DSimpleCropper](#) object.

E3DSimpleCropper.Crop

Crops a point cloud.

```
[VB6]  
void Crop(  
    E3DPointCloud cloudIn,  
    E3DPointCloud cloudOut  
)
```

Parameters

cloudIn

Cloud to be cropped.

cloudOut

Cropped cloud.

E3DSimpleCropper.E3DSimpleCropper

Creates an [E3DSimpleCropper](#) object.

```
[VB6]  
void E3DSimpleCropper(  
)  
  
void E3DSimpleCropper(  
    EFLOAT rangeX,  
    EFLOAT rangeY,  
    EFLOAT rangeZ  
)
```

Parameters

rangeX

Allowed value range along the X axis. Pass NULL if no cropping should be done along this axis.

rangeY

Allowed value range along the Y axis. Pass NULL if no cropping should be done along this axis.

rangeZ

Allowed value range along the Z axis. Pass NULL if no cropping should be done along this axis.

E3DSimpleCropper.XRange

Sets the allowed X value range.

[VB6]

XRange As EFloatRange

read-write

E3DSimpleCropper.YRange

Sets the allowed Y value range.

[VB6]

YRange As EFloatRange

read-write

E3DSimpleCropper.ZRange

Sets the allowed Z value range.

[VB6]

ZRange As EFloatRange

read-write

E3DSphericalCropper Class

Manages a spherical point cloud cropper.

METHODS

Crop

Crops a point cloud.

E3DSphericalCropperCreates an [E3DSphericalCropper](#) object.

3

DSphericalCropper.Crop

Crops a point cloud.

[VB6]

```
void Crop(
    E3DPointCloud cloudIn,
    E3DPointCloud cloudOut,
    Boolean invertCrop
)
```

Parameters

cloudIn

Cloud to be cropped.

cloudOut

Cropped cloud.

invertCrop

Indicates if the points kept must be the points inside (true) or outside (false) the sphere.

E3DSphericalCropper.E3DSphericalCropper

Creates an [E3DSphericalCropper](#) object.

[VB6]

```
void E3DSphericalCropper(
    E3DPoint center,
    Single radius
)
```

Parameters

center

Center of the sphere.

radius

Radius of the sphere.

Easy Class

This class contains static properties and methods specific to the Easy library.

PROPERTIES

[AngleUnit](#)

Current angular unit.

Version	
THODS	<p>M Returns a pointer to a NULL terminated character string that contains the current version number of Open eVision.</p>
CheckLicense	Checks if a given license is available.
CheckLicenses	Check if at least one license is available. Otherwise, an exception is thrown.
CheckOemKey	Checks if the OEM key, if any, matches a given argument.
CloseImageGraphicContext	Releases the device context associated to an image.
FromRadians	Returns the angle, converted from radians to the current angle unit.
GetBestMatchingImageType	Returns the best matching image type for a given file on disk.
GetErrorText	-
OpenImageGraphicContext	Allows to draw in a gray-level or a color image, using any of the Windows device context functions (the image contents is altered, allowing destructive overlays).

[Render3D](#)

Prepares a three dimensional rendering of an image where the gray-level values are taken for altitudes.

[RenderColorHistogram](#)

Prepares a three dimensional rendering of the histogram of a color image: the pixels are drawn in the RGB space rather than in the XY plane.

[Resize](#)

Resizes an image without interpolation

[SetOemKey](#)

Writes an OEM key value on a dongle.

[StartTiming](#)

Starts timing, using the system clock or performance counter.

[StopTiming](#)

Returns the time, in specified time units, elapsed since the last invocation of [Easy::StartTiming](#).

[ToRadians](#)

Returns the angle, converted from current angle unit to radians.

[TrueTimingResolution](#)

Returns the actual resolution of the timing clock, in ticks per seconds.

Easy.AngleUnit

Current angular unit.

[VB6]

AngleUnit As EAngleUnit

read-write

Remarks

All angles are computed using some angular unit, as well on input as on output. The desired unit can be changed at any time. By default, all angles are given in degrees (**0..360**).

Easy.CheckLicense

Checks if a given license is available.

[VB6]

```
Boolean CheckLicense(  
    Features license  
)
```

Parameters

license

The license to check

Easy.CheckLicenses

Check if at least one license is available. Otherwise, an exception is thrown.

[VB6]

```
void CheckLicenses(  
    )
```

Easy.CheckOemKey

Checks if the OEM key, if any, matches a given argument.

[VB6]

```
Boolean CheckOemKey(  
    ()Byte key,  
    Long index  
    )
```

Parameters

key

The expected value of the OEM key

index

The index of the dongle where the OEM key is expected. By default, the first dongle found is selected.

Remarks

The length of the OEM key must be exactly 8 characters.

Easy.CloseImageGraphicContext

Releases the device context associated to an image.

[VB6]

```
void CloseImageGraphicContext(
    EImageBW8 pImage,
    Long hDC
)

void CloseImageGraphicContext(
    EImageC24 pImage,
    Long hDC
)
```

Parameters

pImage

Pointer to the target image (must be the same as that passed to [Easy::OpenImageGraphicContext](#)).

hDC

Handle to a device context that was produced by [Easy::OpenImageGraphicContext](#).

Easy.FromRadians

Returns the angle, converted from radians to the current angle unit.

[VB6]

```
Single FromRadians(
    Single angle
)
```

Parameters

angle

Angle to be converted

Easy.GetBestMatchingImageType

Returns the best matching image type for a given file on disk.

[VB6]

```
EImageType GetBestMatchingImageType(
    String path
)
```

Parameters

path

The path to the file on disk.

Easy.GetErrorText

-

[VB6]

```
String GetErrorText(
    EError error
)
```

Parameters

error

Easy.OpenImageGraphicContext

Allows to draw in a gray-level or a color image, using any of the Windows device context functions (the image contents is altered, allowing destructive overlays).

[VB6]

```

Long OpenImageGraphicContext(
    EImageBW8 pImage
)
Long OpenImageGraphicContext(
    EImageC24 pImage
)

```

Parameters

pImage

Pointer to the target image.

Remarks

The function returns a handle to a device context associated to the image pixel data. When the device context is no more needed, call the [Easy::CloseImageGraphicContext](#) function with the same argument.

Easy.Render3D

Prepares a three dimensional rendering of an image where the gray-level values are taken for altitudes.

[VB6]

```

void Render3D(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Single phi,
    Single psi,
    Single xScale,
    Single yScale,
    Single zScale,
    Long dotSize
)

```

```
void Render3D(
    EROIC24 sourceImage,
    EROIBW8 zImage,
    EROIC24 destinationImage,
    Single phi,
    Single psi,
    Single xScale,
    Single yScale,
    Single zScale,
    Long dotSize
)
```

Parameters

sourceImage

Pointer to the source image.

destinationImage

Pointer to the destination image.

phi

Rotation angle about the X-axis.

psi

Rotation angle about the Y-axis.

xScale

Magnification factor along X (should remain close to **1**).

yScale

Magnification factor along Y (should remain close to **1**).

zScale

Magnification factor along Z (should remain close to **1**).

dotSize

Size of the rendered dots; allowed values are **1, 4, 5 or 9**.

zImage

Pointer to the altitude image.

Remarks

The image is viewed by rotating it about the X-axis, then about the Y-axis. Magnification factors in the three directions (X = width, Y = height, and Z = depth) can be given. The rendered image appears as independent dots. The dot size can be adjusted so that the surface appears more or less opaque. The function does not display the rendered image by itself. Rather, it prepares a destination image to be displayed.

Easy.RenderColorHistogram

Prepares a three dimensional rendering of the histogram of a color image: the pixels are drawn in the RGB space rather than in the XY plane.

[VB6]

```
void RenderColorHistogram(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Single phi,
    Single psi,
    Single xScale,
    Single yScale,
    Single zScale
)

void RenderColorHistogram(
    EROIC24 sourceImage,
    EROIC24 sysImage,
    EROIC24 destinationImage,
    Single phi,
    Single psi,
    Single xScale,
    Single yScale,
    Single zScale
)
```

Parameters

sourceImage

Pointer to the raw source image.

destinationImage

Pointer to the destination image.

phi

Rotation angle about the X-axis.

psi

Rotation angle about the Y-axis.

xScale

Magnification factor along X (should remain close to **1**).

yScale

Magnification factor along Y (should remain close to **1**).

zScale

Magnification factor along Z (should remain close to **1**).

sysImage

Pointer to the source image transformed into another color system.

Remarks

This allows to observe the clustering and dispersion of the RGB values. The image is viewed by rotating it about the X-axis, then about the Y-axis. Magnification factors in the three directions (X = Red, Y = Green, and Z = Blue) can be given. In a more advanced version, prepares a three dimensional rendering of the pixels in another system than RGB (EasyColor provides conversion means). However, the raw RGB image must still be provided to allow the display of the pixels in their usual colors. The rendered image appears as independent dots. The function does not display the rendered image by itself. Rather, it prepares a destination image to be displayed.

Easy.Resize

Resizes an image without interpolation

[VB6]

```
void Resize(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

void Resize(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)

void Resize(
    EROIC15 sourceImage,
    EROIC15 destinationImage
)

void Resize(
    EROIC16 sourceImage,
    EROIC16 destinationImage
)
```

```

void Resize(
    EROIC24 sourceImage,
    EROIC24 destinationImage
)

void Resize(
    EROIC24A sourceImage,
    EROIC24A destinationImage
)

```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

Easy.SetOemKey

Writes an OEM key value on a dongle.

[VB6]

```

void SetOemKey(
    ()Byte key,
    Long index
)

```

Parameters

key

The OEM key value to write

index

The index of the dongle where the OEM key must be written. By default, the first dongle found is selected.

Remarks

The length of the OEM key must be exactly 8 characters. This method raises an [EError_CannotWriteOEMKey](#) error if the value cannot be set properly.

Easy.StartTiming

Starts timing, using the system clock or performance counter.

```
[VB6]  
void StartTiming()  
)
```

Easy.StopTiming

Returns the time, in specified time units, elapsed since the last invocation of [Easy::StartTiming](#).

```
[VB6]  
Long StopTiming(  
    Long resolution  
)
```

Parameters

resolution

Temporal resolution, in ticks per second.

Easy.ToRadians

Returns the angle, converted from current angle unit to radians.

```
[VB6]
```

```
Single ToRadians(  
    Single angle  
)
```

Parameters

angle

Angle to be converted

Easy.TrueTimingResolution

Returns the actual resolution of the timing clock, in ticks per seconds.

[VB6]

```
Long TrueTimingResolution(  
)
```

Remarks

Timing granularity is hardware-dependent, but is usually better than 1 µs. This function can be used to select an appropriate timing resolution when using [Easy::StopTiming](#).

Easy.Version

Returns a pointer to a **NULL** terminated character string that contains the current version number of Open eVision.

[VB6]

```
Version As String  
read-only
```

EasyColor Class

This class contains static properties and methods specific to the EasyColor library.

PROPERTIES

CieAB

-

CieAG

-

CieAR

-

CieD50B

-

CieD50G

-

CieD50R

-

CieD55B

-

CieD55G

-

CieD55R

-

CieD65B	-
CieD65G	-
CieD65R	-
CieFB	-
CieFG	-
CieFR	-
CompensateNtscGamma	-
CompensatePalGamma	-
CompensateSmpteGamma	-
DstQuantization	Quantization mode for output values.
NtscGamma	-
PalGamma	-

RgbStandard	RGB definition to be used when converting between RGB and other color systems.
SmpteGamma	-
SrcQuantization	M Quantization mode for input values.
THODS	E
AssignNearestClass	Assigns to every pixel of the source image the nearest class index <i>plus one</i> and stores its value in the destination image.
AssignNearestClassCenter	Assigns to every pixel of the source image the nearest class center and stores its value in the destination image.
BayerToC24	Converts a Bayer pattern encoded image into a color image.
C24ToBayer	Converts a color image into a Bayer pattern encoded image.
ClassAverages	Computes the average source pixel colors for every class separately.

[ClassVariances](#)

Computes the averages and variances of the image pixel colors for every class separately.

[Compose](#)

Combines three gray-level images, considered as three color planes, into a color image.

[Decompose](#)

Extracts the three color planes, considered as gray-level images, from a color image.

[Dequantize](#)

Convert a quantized value to a unquantized value of a given color system.

[Format422To444](#)

Converts a YUV 4:2:2 image to a YUV 4:4:4 image using interpolation.

[Format444To422](#)

Converts a YUV 4:4:4 image to a YUV 4:2:2 image using filtering

[GetComponent](#)

Extracts one color plane, considered as a gray-level image, from a color image.

[ImproveClassCenters](#)

Redefines the class centers by computing the average color value of the pixels assigned to each class in the source image.

IshToRgb	Convert a color from any system to RGB.
LabToRgb	Convert a color from any system to RGB.
LabToXyz	Convert a color from one system to another, including the xyz variant (reduced XYZ).
LchToRgb	Convert a color from any system to RGB.
LshToRgb	Convert a color from any system to RGB.
LuvToRgb	Convert a color from any system to RGB.
LuvToXyz	Convert a color from one system to another, including the xyz variant (reduced XYZ).
PseudoColor	Applies the mapping defined by the pseudo-color lookup table to transform a gray-level image into a color image.
Quantize	Convert an unquantized color of a given color system to a quantized color.
RegisterPlanes	Sets a color plane of a color image by using a gray-level image as component.

RgbToLsh	Convert a color from RGB to another system.
RgbToLab	Convert a color from RGB to another system.
RgbToLch	Convert a color from RGB to another system.
RgbToLsh	Convert a color from RGB to another system.
RgbToLuv	Convert a color from RGB to another system.
RgbToReducedXyz	Convert a color from one system to another, including the xyz variant (reduced XYZ).
RgbToVsh	Convert a color from RGB to another system.
RgbToXyz	Convert a color from RGB to another system.
RgbToYiq	Convert a color from RGB to another system.
RgbToYsh	Convert a color from RGB to another system.
RgbToYuv	Convert a color from RGB to another system.
SetComponent	Sets a color plane of a color image by using a gray-level image as component.

Transform	Applies a color transformation to a specified image.
TransformBayer	Converts an image, using the transformation defined by a color lookup.
VshToRgb	Convert a color from any system to RGB.
XyzToLab	Convert a color from one system to another, including the xyz variant (reduced XYZ).
XyzToLuv	Convert a color from one system to another, including the xyz variant (reduced XYZ).
XyzToRgb	Convert a color from any system to RGB.
YiqToRgb	Convert a color from any system to RGB.
YshToRgb	Convert a color from any system to RGB.
YuvToRgb	Convert a color from any system to RGB.

EasyColor.AssignNearestClass

Assigns to every pixel of the source image the nearest class index *plus one* and stores its value in the destination image.

[VB6]

```
void AssignNearestClass(
    EROIC24 sourceImage,
    EROIBW8 destinationImage,
    EC24Vector classCenters
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination gray-level image/ROI.

classCenters

Pointer to the vector of the class centers.

Remarks

This generates a labeled gray-level image for use with EasyObject (see [EImageEncoder](#) and [ELabeledImageSegmenter](#)).

Note. The class index plus one is stored instead of the class index because EasyObject will never code class 0 objects.

Color image segmentation allows you to decompose a color image in different regions by assigning a "class" (integer index) to every pixel. The nearest neighbor method is used, i.e. for each class a representative center is specified, and a given pixel is associated to the class with the closest center. Color image segmentation can be used in conjunction with EasyObject to perform blob analysis on the segmented regions. To use the color segmentation functions, the set of class centers must be specified as a vector of [EC24](#) elements. In this sense, the method is termed supervised clustering. A good way to obtain these values is to compute the average color in an ROI. Unsupervised clustering is also made available by implementing the so called K-means method that automatically improves an initial choice of class centers.

EasyColor.AssignNearestClassCenter

Assigns to every pixel of the source image the nearest class center and stores its value in the destination image.

[VB6]

```
void AssignNearestClassCenter(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    EC24Vector classCenters
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

classCenters

Pointer to the vector of the class centers.

Remarks

This generates a labeled color image. Color image segmentation allows you to decompose a color image in different regions by assigning a "class" (integer index) to every pixel. The nearest neighbor method is used, i.e. for each class a representative center is specified, and a given pixel is associated to the class with the closest center. Color image segmentation can be used in conjunction with EasyObject to perform blob analysis on the segmented regions. To use the color segmentation functions, the set of class centers must be specified as a vector of [EC24](#) elements. In this sense, the method is termed supervised clustering. A good way to obtain these values is to compute the average color in an ROI. Unsupervised clustering is also made available by implementing the so called K-means method that automatically improves an initial choice of class centers.

EasyColor.BayerToC24

Converts a Bayer pattern encoded image into a color image.

[VB6]

```
void BayerToC24(
    EROIBW8 sourceImage,
    EROIC24 destinationImage,
    Boolean evenCol,
    Boolean evenRow,
    Boolean interpolate,
    Boolean improved
)
```

Parameters

sourceImage

Pointer to the Bayer pattern input image/ROI, stored using the 8 bits per pixel format.

destinationImage

Pointer to the color output image/ROI.

evenCol

TRUE if the leftmost image column contains no blue pixels.

evenRow

TRUE if the topmost image row contains no red pixels.

interpolate

Interpolation mode to be used for pixel reconstruction. When **FALSE**, the missing color components are merely copied from northern/western pixels; when **TRUE**, they are computed by averaging from relevant neighbors. By default, interpolation is used.

improved

Provides an access to an improved interpolation mode that reduces visible artifacts along object edges. The running time of the improved method is longer. By default, it is not used.

Remarks

The pattern can be shifted by one pixel horizontally and vertically as needed to deal with a non standard pattern origin. See also Bayer Filter.

EasyColor.C24ToBayer

Converts a color image into a Bayer pattern encoded image.

```
[VB6]  
  
void C24ToBayer(  
    EROIC24 sourceImage,  
    EROIBW8 destinationImage,  
    Boolean evenCol,  
    Boolean evenRow  
)
```

Parameters

sourceImage

Pointer to the color input image/ROI.

destinationImage

Pointer to the Bayer pattern output image/ROI, stored using the 8 bits per pixel format.

evenCol

TRUE if the leftmost destination image column can't contain blue pixels.

evenRow

TRUE if the topmost destination image row can't contain red pixels.

Remarks

The pattern can be shifted by one pixel horizontally and vertically as needed to deal with a non standard pattern origin. See also Bayer Filter.

EasyColor.CieAB

CieAB As Single

read-only

EasyColor.CieAG

[VB6]

CieAG As Single

read-only

EasyColor.CieAR

-

[VB6]

CieAR As Single

read-only

EasyColor.CieD50B

-

[VB6]

CieD50B As Single

read-only

EasyColor.CieD50G

-

[VB6]

CieD50G As Single

read-only

EasyColor.CieD50R

-

[VB6]

CieD50R As Single

read-only

EasyColor.CieD55B

-

[VB6]

CieD55B As Single

read-only

EasyColor.CieD55G

-

[VB6]

CieD55G As Single

read-only

EasyColor.CieD55R

-

[VB6]

CieD55R As Single

read-only

EasyColor.CieD65B

-

[VB6]

CieD65B As Single

read-only

EasyColor.CieD65G

-

[VB6]

CieD65G As Single

read-only

EasyColor.CieD65R

-

[VB6]

CieD65R As Single

read-only

EasyColor.CieFB

-

[VB6]

CieFB As Single

read-only

EasyColor.CieFG

-

[VB6]

CieFG As Single

read-only

EasyColor.CieFR

-

[VB6]

CieFR As Single

read-only

EasyColor.ClassAverages

Computes the average source pixel colors for every class separately.

[VB6]

```
void ClassAverages(
    EROIC24 sourceImage,
    EC24Vector classCenters,
    EColorVector averages
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

classCenters

Pointer to the vector of the class centers.

averages

Pointer to the vector of the average color values.

Remarks

This allows measuring the actual average color of the segmented regions. Color image segmentation allows you to decompose a color image in different regions by assigning a "class" (integer index) to every pixel. The nearest neighbor method is used, i.e. for each class a representative center is specified, and a given pixel is associated to the class with the closest center. Color image segmentation can be used in conjunction with EasyObject to perform blob analysis on the segmented regions. To the color segmentation functions, the set of class centers must be specified as a vector of EC24 elements. In this sense, the method is termed supervised clustering. A good way to obtain these values is to compute the average color in an ROI. Unsupervised clustering is also made available by implementing the so called K-means method that automatically improves an initial choice of class centers.

EasyColor.ClassVariances

Computes the averages and variances of the image pixel colors for every class separately.

[VB6]

```
void ClassVariances(
    EROIC24 sourceImage,
    EC24Vector classCenters,
    EColorVector averages,
    EColorVector variances
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

classCenters

Pointer to the vector of the class centers.

averages

Pointer to the vector of the average color values.

variances

Pointer to the vector of the variance color values.

Remarks

This allows quantifying the homogeneity of the segmented regions. Color image segmentation allows you to decompose a color image in different regions by assigning a "class" (integer index) to every pixel. The nearest neighbor method is used, i.e. for each class a representative center is specified, and a given pixel is associated to the class with the closest center. Color image segmentation can be used in conjunction with EasyObject to perform blob analysis on the segmented regions. To the color segmentation functions, the set of class centers must be specified as a vector of EC24 elements. In this sense, the method is termed supervised clustering. A good way to obtain these values is to compute the average color in an ROI. Unsupervised clustering is also made available by implementing the so called K-means method that automatically improves an initial choice of class centers.

EasyColor.CompensateNtscGamma

-

[VB6]

CompensateNtscGamma As Single

read-only

EasyColor.CompensatePalGamma

-

[VB6]

CompensatePalGamma As Single

read-only

EasyColor.CompensateSmpteGamma

-

[VB6]

CompensateSmpTEGamma As Single

read-only

EasyColor.Compose

Combines three gray-level images, considered as three color planes, into a color image.

[VB6]

```
void Compose(
    EROIBW8 sourceImageOfColor0,
    EROIBW8 sourceImageOfColor1,
    EROIBW8 sourceImageOfColor2,
    EROIC24 colorDestinationImage,
    EColorLookup lookup
)

void Compose(
    EROIBW16 sourceImageOfColor0,
    EROIBW16 sourceImageOfColor1,
    EROIBW16 sourceImageOfColor2,
    EROIC48 colorDestinationImage
)
```

Parameters

sourceImageOfColor0

Pointers to the three input gray-level component images/ROIs.

sourceImageOfColor1

Pointers to the three input gray-level component images/ROIs.

sourceImageOfColor2

Pointers to the three input gray-level component images/ROIs.

colorDestinationImage

Pointer to the output color image/ROI.

lookup

Pointer to the color lookup table, or **NULL**.

Remarks

If a color lookup is used, the resulting image undergoes the corresponding color transform. This way, it is possible to build an RGB image from the color planes of another system, or conversely. A color image can be seen as a set of three color planes, each corresponding to a color component. The color planes are themselves continuous tone images. An [EImageC24](#) contains three [EImageBW8](#).

EasyColor.Decompose

Extracts the three color planes, considered as gray-level images, from a color image.

[VB6]

```
void Decompose(
    EROIC24 colorSourceImage,
    EROIBW8 destinationImageOfColor0,
    EROIBW8 destinationImageOfColor1,
    EROIBW8 destinationImageOfColor2,
    EColorLookup lookup
)

void Decompose(
    EROIC48 colorSourceImage,
    EROIBW16 destinationImageOfColor0,
    EROIBW16 destinationImageOfColor1,
    EROIBW16 destinationImageOfColor2
)
```

Parameters

colorSourceImage

Pointer to the input color image/ROI.

destinationImageOfColor0

Pointers to the three output gray level component images/ROIs.

destinationImageOfColor1

Pointers to the three output gray level component images/ROIs.

destinationImageOfColor2

Pointers to the three output gray level component images/ROIs.

lookup

Pointer to the color lookup table, or **NULL**.

Remarks

If a color lookup is used, the source image undergoes the corresponding color transform. This way, it is possible to get the RGB components from an image of another system, or conversely. A color image can be seen as a set of three color planes, each corresponding to a color component. The color planes are themselves continuous tone images. An [EImageC24](#) contains three [EImageBW8](#).

EasyColor.Dequantize

Convert a quantized value to a unquantized value of a given color system.

[VB6]

```
void Dequantize(
    EC24 colorIn,
    ERGB colorOut
)

void Dequantize(
    EC24 colorIn,
    EXYZ colorOut
)

void Dequantize(
    EC24 colorIn,
    EYUV colorOut
)

void Dequantize(
    EC24 colorIn,
    EYIQ colorOut
)

void Dequantize(
    EC24 colorIn,
    ELSH colorOut
)

void Dequantize(
    EC24 colorIn,
    EVSH colorOut
)
```

```
void Dequantize(
    EC24 colorIn,
    EISH colorOut
)

void Dequantize(
    EC24 colorIn,
    EYSH colorOut
)

void Dequantize(
    EC24 colorIn,
    ELAB colorOut
)

void Dequantize(
    EC24 colorIn,
    ELCH colorOut
)

void Dequantize(
    EC24 colorIn,
    ELUV colorOut
)
```

Parameters

colorIn

Input quantized color.

colorOut

Output unquantized color, as defined by the corresponding structure.

Remarks

Quantization is the process that transforms a continuous value, usually represented as a floating-point value in the [0..1] interval, into a discrete one, usually represented as an integer in the [0..255] interval. Dequantization is the reverse process.

EasyColor.DstQuantization

Quantization mode for output values.

[VB6]

DstQuantization As EColorQuantization

read-write

Remarks

These settings remain permanent and influence the relevant quantized and unquantized color conversions (during lookup table initialization or image color transformation).

EasyColor.Format422To444

Converts a YUV 4:2:2 image to a YUV 4:4:4 image using interpolation.

[VB6]

```
void Format422To444(
    EROIBW16 sourceImage,
    EROIC24 destinationImage,
    Boolean yFirst
)
```

Parameters

sourceImage

Pointer to the input image/ROI, stored using the 16 bits per pixel format.

destinationImage

Pointer to the output image/ROI.

yFirst

Flag indicating if the format is YUYVYUYV (**TRUE**) or UYVYUYVY (**FALSE**).

Remarks

In the YUV system, it has been established that sub-sampling the chroma components does not degrade the visual image quality. The 4:4:4 format uses three bytes of information by pixel. The 4:2:2 format is such that only the U and V chroma of the even pixels are kept and they are stored with the even and odd luma, as follows: Thus, only two bytes per pixel are required. $Y_{[\text{even}]} \ U_{[\text{even}]} \ Y_{[\text{odd}]} \ V_{[\text{even}]}$

EasyColor.Format444To422

Converts a YUV 4:4:4 image to a YUV 4:2:2 image using filtering

[VB6]

```
void Format444To422(
    EROIC24 sourceImage,
    EROIBW16 destinationImage,
    Boolean yFirst
)
```

Parameters

sourceImage

Pointer to the input image/ROI.

destinationImage

Pointer to the output image/ROI, stored using the 16 bits per pixel format.

yFirst

Flag indicating if the format is YUVYVUYV (**TRUE**) or UYVYUYVY (**FALSE**).

Remarks

In the YUV system, it has been established that sub-sampling the chroma components does not degrade the visual image quality. The 4:4:4 format uses three bytes of information by pixel. The 4:2:2 format is such that only the U and V chroma of the even pixels are kept and they are stored with the even and odd luma, as follows: Thus, only two bytes per pixel are required. $Y_{[\text{even}]} \ U_{[\text{even}]} \ Y_{[\text{odd}]} \ V_{[\text{even}]}$

EasyColor.GetComponent

Extracts one color plane, considered as a gray-level image, from a color image.

[VB6]

```

void GetComponent(
    EROIC24 colorSourceImage,
    EROIBW8 bWDestinationImage,
    Long component,
    EColorLookup lookup
)

void GetComponent(
    EROIC48 colorSourceImage,
    EROIBW16 bWDestinationImage,
    Long component
)

```

Parameters

colorSourceImage

Pointer to the input color image/ROI.

bWDestinationImage

Pointers to the output gray-level component image/ROI.

component

Color component index (0, 1, or 2).

lookup

Pointer to the color lookup table, or **NULL**.

EasyColor.ImproveClassCenters

Redefines the class centers by computing the average color value of the pixels assigned to each class in the source image.

[VB6]

```

void ImproveClassCenters(
    EROIC24 sourceImage,
    EC24Vector classCenters
)

```

Parameters

sourceImage

Pointer to the source image/ROI.

classCenters

Pointer to the vector of the class centers.

Remarks

This implements a step of the K-means method for unsupervised clustering. Color image segmentation allows you to decompose a color image in different regions by assigning a "class" (integer index) to every pixel. The nearest neighbor method is used, i.e. for each class a representative center is specified, and a given pixel is associated to the class with the closest center. Color image segmentation can be used in conjunction with EasyObject to perform blob analysis on the segmented regions.

EasyColor.IshToRgb

Convert a color from any system to RGB.

[VB6]

```
void IshToRgb(
    EISH colorIn,
    ERGB colorOut
)

void IshToRgb(
    EC24 colorIn,
    EC24 colorOut
)
```

Parameters

colorIn

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system.

EasyColor.LabToRgb

Convert a color from any system to RGB.

```
[VB6]  
void LabToRgb(  
    ELAB colorIn,  
    ERGB colorOut  
)  
  
void LabToRgb(  
    EC24 colorIn,  
    EC24 colorOut  
)
```

Parameters

colorIn

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system.

EasyColor.LabToXyz

Convert a color from one system to another, including the xyz variant (reduced XYZ).

```
[VB6]  
void LabToXyz(  
    ELAB colorIn,  
    EXYZ colorOut  
)
```

```
void LabToXyz(
    EC24 colorIn,
    EC24 colorOut
)
```

Parameters

colorIn
Input color.
colorOut
Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system.

EasyColor.LchToRgb

Convert a color from any system to RGB.

[VB6]

```
void LchToRgb(
    ELCH colorIn,
    ERGB colorOut
)

void LchToRgb(
    EC24 colorIn,
    EC24 colorOut
)
```

Parameters

colorIn
Input color.
colorOut
Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system.

EasyColor.LshToRgb

Convert a color from any system to RGB.

```
[VB6]  
void LshToRgb(  
    ELSH colorIn,  
    ERGB colorOut  
)  
  
void LshToRgb(  
    EC24 colorIn,  
    EC24 colorOut  
)
```

Parameters

colorIn

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system.

EasyColor.LuvToRgb

Convert a color from any system to RGB.

```
[VB6]
```

```

void LuvToRgb(
    ELUV colorIn,
    ERGB colorOut
)

void LuvToRgb(
    EC24 colorIn,
    EC24 colorOut
)

```

Parameters*colorIn*

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system.

EasyColor.LuvToXyz

Convert a color from one system to another, including the xyz variant (reduced XYZ).

[VB6]

```

void LuvToXyz(
    ELUV colorIn,
    EXYZ colorOut
)

void LuvToXyz(
    EC24 colorIn,
    EC24 colorOut
)

```

Parameters*colorIn*

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system.

EasyColor.NtscGamma

-

[VB6]

NtscGamma As Single

read-only

EasyColor.PalGamma

-

[VB6]

PalGamma As Single

read-only

EasyColor.PseudoColor

Applies the mapping defined by the pseudo-color lookup table to transform a gray-level image into a color image.

[VB6]

```
void PseudoColor(
    EROIBW8 sourceImage,
    EROIC24 destinationImage,
    EPseudoColorLookup lookup
)
```

Parameters

sourceImage

Pointer to the source gray-level image.

destinationImage

Pointer to the destination color image.

lookup

Pointer to the pseudo-color lookup table.

Remarks

Pseudo-coloring is a convenient way to display gray-level images with enhanced contrast: a shade of colors is associated to the shade of gray-level values. A simple way to define the shade of colors is to specify a path in color space. In order to use pseudo-coloring, a special lookup table is used: [EPseudoColorLookup](#). It handles the mapping between the gray-level and color values.

EasyColor.Quantize

Convert an unquantized color of a given color system to a quantized color.

[VB6]

```
void Quantize(
    ERGB colorIn,
    EC24 colorOut
)

void Quantize(
    EXYZ colorIn,
    EC24 colorOut
)
```

```
void Quantize(
    EYUV colorIn,
    EC24 colorOut
)

void Quantize(
    EYIQ colorIn,
    EC24 colorOut
)

void Quantize(
    ELSH colorIn,
    EC24 colorOut
)

void Quantize(
    EVSH colorIn,
    EC24 colorOut
)

void Quantize(
    EISH colorIn,
    EC24 colorOut
)

void Quantize(
    EYSH colorIn,
    EC24 colorOut
)

void Quantize(
    ELAB colorIn,
    EC24 colorOut
)

void Quantize(
    ELCH colorIn,
    EC24 colorOut
)

void Quantize(
    ELUV colorIn,
    EC24 colorOut
)
```

Parameters

colorIn

Input unquantized color, as defined by the corresponding structure.

colorOut

Output quantized color.

Remarks

Quantization is the process that transforms a continuous value, usually represented as a floating-point value in the [0..1] interval, into a discrete one, usually represented as an integer in the [0..255] interval. Dequantization is the reverse process.

EasyColor.RegisterPlanes

Sets a color plane of a color image by using a gray-level image as component.

[VB6]

```
void RegisterPlanes(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Long rShiftX,
    Long gShiftX,
    Long bShiftX,
    Long rShiftY,
    Long gShiftY,
    Long bShiftY
)
```

Parameters

sourceImage

Pointers to the input image/ROI.

destinationImage

Pointer to the output image/ROI.

rShiftX

Horizontal shifting of the first plane (the red one in case of an RGB image), expressed in pixels.

gShiftX

Horizontal shifting of the second plane (the green one in case of an RGB image), expressed in pixels.

bShiftX

Horizontal shifting of the third plane (the blue one in case of an RGB image), expressed in pixels.

rShiftY

Vertical shifting of the first plane (the red one in case of an RGB image), expressed in pixels.

gShiftY

Vertical shifting of the second plane (the green one in case of an RGB image), expressed in pixels.

bShiftY

Vertical shifting of the third plane (the blue one in case of an RGB image), expressed in pixels.

Remarks

A color image can be seen as a set of three color planes, each corresponding to a color component. The color planes are themselves continuous tone images. An [ElImageC24](#) contains three [ElImageBW8](#).

EasyColor.RgbStandard

RGB definition to be used when converting between RGB and other color systems.

[VB6]

```
RgbStandard As ERgbStandard
```

read-write

Remarks

Some variant of the color systems can be used. The [EasyColor::SrcQuantization](#) and [EasyColor::DstQuantization](#) functions are used to activate them. These settings remain permanent and influence the relevant quantized and unquantized color conversions (during lookup table initialization or image color transformation).

EasyColor.RgbTolsh

Convert a color from RGB to another system.

[VB6]

```

void RgbToIsh(
    ERGB colorIn,
    EISH colorOut
)

void RgbToIsh(
    EC24 colorIn,
    EC24 colorOut
)

```

Parameters

colorIn
 Input color.
colorOut
 Output color, as defined by the corresponding structure.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system.

EasyColor.RgbToLab

Convert a color from RGB to another system.

[VB6]

```

void RgbToLab(
    ERGB colorIn,
    ELAB colorOut
)

void RgbToLab(
    EC24 colorIn,
    EC24 colorOut
)

```

Parameters

colorIn
 Input color.
colorOut

Output color, as defined by the corresponding structure.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system.

EasyColor.RgbToLch

Convert a color from RGB to another system.

```
[VB6]  
  
void RgbToLch(  
    ERGB colorIn,  
    ELCH colorOut  
)  
  
void RgbToLch(  
    EC24 colorIn,  
    EC24 colorOut  
)
```

Parameters

colorIn

Input color.

colorOut

Output color, as defined by the corresponding structure.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system.

EasyColor.RgbToLsh

Convert a color from RGB to another system.

[VB6]

```
void RgbToLsh(
    ERGB colorIn,
    ELSH colorOut
)

void RgbToLsh(
    EC24 colorIn,
    EC24 colorOut
)
```

Parameters

colorIn

Input color.

colorOut

Output color, as defined by the corresponding structure.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system.

EasyColor.RgbToLuv

Convert a color from RGB to another system.

[VB6]

```
void RgbToLuv(
    ERGB colorIn,
    ELUV colorOut
)

void RgbToLuv(
    EC24 colorIn,
    EC24 colorOut
)
```

Parameters

colorIn

Input color.

colorOut

Output color, as defined by the corresponding structure.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system.

EasyColor.RgbToReducedXyz

Convert a color from one system to another, including the xyz variant (reduced XYZ).

[VB6]

```
void RgbToReducedXyz (
    ERGB colorIn,
    EXYZ colorOut
)

void RgbToReducedXyz (
    EC24 colorIn,
    EC24 colorOut
)
```

Parameters

colorIn

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system.

EasyColor.RgbToVsh

Convert a color from RGB to another system.

[VB6]

```
void RgbToVsh(
    ERGB colorIn,
    EVSH colorOut
)

void RgbToVsh(
    EC24 colorIn,
    EC24 colorOut
)
```

Parameters

colorIn

Input color.

colorOut

Output color, as defined by the corresponding structure.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system.

EasyColor.RgbToXyz

Convert a color from RGB to another system.

[VB6]

```
void RgbToXyz(
    ERGB colorIn,
    EXYZ colorOut
)

void RgbToXyz(
    EC24 colorIn,
    EC24 colorOut
)
```

Parameters

colorIn

Input color.

colorOut

Output color, as defined by the corresponding structure.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system.

EasyColor.RgbToYiq

Convert a color from RGB to another system.

[VB6]

```
void RgbToYiq(
    ERGB colorIn,
    EYIQ colorOut
)

void RgbToYiq(
    EC24 colorIn,
    EC24 colorOut
)
```

Parameters

colorIn

Input color.

colorOut

Output color, as defined by the corresponding structure.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system.

EasyColor.RgbToYsh

Convert a color from RGB to another system.

```
[VB6]
void RgbToYsh(
    ERGB colorIn,
    EYSH colorOut
)

void RgbToYsh(
    EC24 colorIn,
    EC24 colorOut
)
```

Parameters

colorIn

Input color.

colorOut

Output color, as defined by the corresponding structure.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system.

EasyColor.RgbToYuv

Convert a color from RGB to another system.

```
[VB6]
void RgbToYuv(
    ERGB colorIn,
    EYUV colorOut
)

void RgbToYuv(
    EC24 colorIn,
    EC24 colorOut
)
```

Parameters

colorIn

Input color.

colorOut

Output color, as defined by the corresponding structure.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system.

EasyColor.SetComponent

Sets a color plane of a color image by using a gray-level image as component.

[VB6]

```
void SetComponent(
    EROIBW8 bWSOURCEImage,
    EROIC24 colorDestinationImage,
    Long component
)

void SetComponent(
    EROIBW16 bWSOURCEImage,
    EROIC48 colorDestinationImage,
    Long component
)
```

Parameters

bWSOURCEImage

Pointers to the input gray level component image/ROI.

colorDestinationImage

Pointer to the output color image/ROI.

component

Color component index (**0**, **1**, or **2**).

Remarks

A color image can be seen as a set of three color planes, each corresponding to a color component. The color planes are themselves continuous tone images. An [EImageC24](#) contains three [EImageBW8](#).

EasyColor.SmpteGamma

-

[VB6]

SmpteGamma As Single

read-only

EasyColor.SrcQuantization

Quantization mode for input values.

[VB6]

SrcQuantization As EColorQuantization

read-write

Remarks

These settings remain permanent and influence the relevant quantized and unquantized color conversions (during lookup table initialization or image color transformation).

EasyColor.Transform

Applies a color transformation to a specified image.

[VB6]

```
void Transform(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    EColorLookup lookup
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

lookup

Pointer to the color lookup.

Remarks

In the first case, the transformation is defined by a color lookup. See Initialization ([EColorLookup](#)). In the two other cases, the user defines a quantized or unquantized color transformation. No intermediate color lookup table is used. A color image can be seen as a set of three color planes, each corresponding to a color component. The color planes are themselves continuous tone images. An [EImageC24](#) contains three [EImageBW8](#).

EasyColor.TransformBayer

Converts an image, using the transformation defined by a color lookup.

[VB6]

```
void TransformBayer(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    EColorLookup lookup,
    Boolean evenCol,
    Boolean evenRow
)
```

Parameters

sourceImage

Pointer to the source image/ROI. This image must be encoded using the Bayer color pattern.

destinationImage

Pointer to the destination image/ROI. This image must be encoded using the Bayer color pattern.

lookup

Pointer to the color lookup table holding the color adjustment transform. The lookup table must be previously set up by [EColorLookup::WhiteBalance](#) method (no other transforms are supported).

evenCol

TRUE if the leftmost destination image column can't contain blue pixels.

evenRow

TRUE if the topmost destination image row can't contain red pixels.

Remarks

By contrast with [EasyColor::Transform](#), the transformation is applied directly to Bayer-encoded data. This allows efficient processing to take place before conversion to the **C24** format.

EasyColor.VshToRgb

Convert a color from any system to RGB.

[VB6]

```
void VshToRgb(
    EVSH colorIn,
    ERGB colorOut
)

void VshToRgb(
    EC24 colorIn,
    EC24 colorOut
)
```

Parameters

colorIn

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system.

EasyColor.XyzToLab

Convert a color from one system to another, including the xyz variant (reduced XYZ).

```
[VB6]  
void XyzToLab(  
    EXYZ colorIn,  
    ELAB colorOut  
)  
  
void XyzToLab(  
    EC24 colorIn,  
    EC24 colorOut  
)
```

Parameters

colorIn

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system.

EasyColor.XyzToLuv

Convert a color from one system to another, including the xyz variant (reduced XYZ).

```
[VB6]
```

```

void XyzToLuv(
    EXYZ colorIn,
    ELUV colorOut
)

void XyzToLuv(
    EC24 colorIn,
    EC24 colorOut
)

```

Parameters*colorIn*

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system.

EasyColor.XyzToRgb

Convert a color from any system to RGB.

[VB6]

```

void XyzToRgb(
    EXYZ colorIn,
    ERGB colorOut
)

void XyzToRgb(
    EC24 colorIn,
    EC24 colorOut
)

```

Parameters*colorIn*

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system.

EasyColor.YiqToRgb

Convert a color from any system to RGB.

[VB6]

```
void YiqToRgb(
    EYIQ colorIn,
    ERGB colorOut
)

void YiqToRgb(
    EC24 colorIn,
    EC24 colorOut
)
```

Parameters

colorIn

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system.

EasyColor.YshToRgb

Convert a color from any system to RGB.

[VB6]

```
void YshToRgb(
    EYSH colorIn,
    ERGB colorOut
)

void YshToRgb(
    EC24 colorIn,
    EC24 colorOut
)
```

Parameters

colorIn

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system.

EasyColor.YuvToRgb

Convert a color from any system to RGB.

[VB6]

```
void YuvToRgb(
    EYUV colorIn,
    ERGB colorOut
)

void YuvToRgb(
    EC24 colorIn,
    EC24 colorOut
)
```

Parameters

colorIn

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system.

EasylImage Class

This class contains static properties and methods specific to the EasylImage library.

PROPERTIES

[OverlayColor](#)

M Sets the color of the overlay in the destination image when a **BW8** Image is used as overlay source image in functions:

THODS

[AdaptiveThreshold](#)

Performs a locally adaptive threshold on the source image.

[AnalyseHistogram](#)

Returns a floating-point statistical parameter extracted from a range of gray levels in an image histogram (most/least frequent value/frequency, min/max value, count, average, standard deviation).

[AnalyseHistogramBW16](#)

Returns a floating-point statistical parameter extracted from a range of gray levels in an image histogram (most/least frequent value/frequency, min/max value, count, average, standard deviation).

[Area](#)

[AreaDoubleThreshold](#)

Counts the pixels whose values are above (or on) a threshold.

[ArgumentImage](#)

Prepares a lookup-table image for use for gradient argument computation.

[AutoThreshold](#)

Returns a suitable threshold value for a gray-level image binarization.

[BiLevelBlackTopHatBox](#)

Performs a top-hat filtering on a bilevel image (closed image minus source image) on a rectangular kernel.

[BiLevelBlackTopHatDisk](#)

Performs a top-hat filtering on a bilevel image (closed image minus source image) on a quasi-circular kernel.

[BiLevelCloseBox](#)

Performs a closing on a bilevel image (dilation followed by erosion) on a rectangular kernel.

[BiLevelCloseDisk](#)

Performs a closing on a bilevel image (dilation followed by erosion) on a quasi-circular kernel.

[BiLevelDilateBox](#)

Performs a dilation on a bilevel image (maximum of the pixel values in a defined neighborhood) on a rectangular kernel. For bilevel images, this maximum can either be 0 (if all pixels are black in the given neighborhood), or the maximum possible pixel value.

[BiLevelDilateDisk](#)

Performs a dilation on a bilevel image (maximum of the pixel values in a defined neighborhood) on a quasi-circular kernel. For bilevel images, this maximum can either be 0 (if all pixels are black in the given neighborhood), or the maximum possible pixel value.

[BiLevelErodeBox](#)

Performs an erosion on a bilevel image (minimum of the pixel values in a defined neighborhood) on a rectangular kernel. For bilevel images, this minimum can either be 0, or the maximum possible pixel value (if all pixels are white in the given neighborhood)

[BiLevelErodeDisk](#)

Performs an erosion on a bilevel image (minimum of the pixel values in a defined neighborhood) on a quasi-circular kernel. For bilevel images, this minimum can either be 0, or the maximum possible pixel value (if all pixels are white in the given neighborhood)

BiLevelMedian	Applies a median filter to a bilevel image (median of the gray values in a 3x3 neighborhood).
BiLevelMorphoGradientBox	Computes the morphological gradient of a bilevel image using a rectangular kernel.
BiLevelMorphoGradientDisk	Computes the morphological gradient of a bilevel image using a quasi-circular kernel.
BiLevelOpenBox	Performs an opening on a bilevel image (erosion followed by dilation) on a rectangular kernel.
BiLevelOpenDisk	Performs an opening on a bilevel image (erosion followed by dilation) on a quasi-circular kernel.
BiLevelThick	Applies a thickening operation on a bilevel image, using a 3x3 kernel.
BiLevelThin	Applies a thinning operation on a bilevel image, using a 3x3 kernel.
BiLevelWhiteTopHatBox	Performs a top-hat filtering on a bilevel image (source image minus opened image) on a rectangular kernel.
BiLevelWhiteTopHatDisk	Performs a top-hat filtering on a bilevel image (source image minus opened image) on a quasi-circular kernel.

[BinaryMoments](#)

Computes the zero-th, first or second order moments on the binarized image, i.e. with a unit weight for those pixels with a value above or equal to the threshold, and zero otherwise.

[BlackTopHatBox](#)

Performs a top-hat filtering on an image (closed image minus source image) on a rectangular kernel.

[BlackTopHatDisk](#)

Performs a top-hat filtering on an image (closed image minus source image) on a quasi-circular kernel.

[CloseBox](#)

Performs a closing on an image (dilation followed by erosion) on a rectangular kernel.

[CloseDisk](#)

Performs a closing on an image (dilation followed by erosion) on a quasi-circular kernel.

[Contour](#)

Follows the *contour* of an object.

[Convert](#)

Transforms the contents of an image to an image of another type.

[ConvertTo422](#)

Turns an 8-bit gray-level image into a YUV 4:2:2 encoded color image.

ConvolGaussian	Applies Gaussian filtering (binomial weights) in rectangular kernel of odd size.
ConvolGradient	Extracts the edges of an image by summing the absolute values of the Gradient X and Gradient Y derivatives and stores the absolute value (magnitude) of the result in the destination image.
ConvolGradientX	Derives an image along X using a Gradient kernel and stores the absolute value (magnitude) of the result in the destination image.
ConvolGradientY	Derives an image along Y using a Gradient kernel and stores the absolute value (magnitude) of the result in the destination image.
ConvolHighpass1	Filters an image using a 3x3 high-pass kernel and stores the absolute value (magnitude) of the result in the destination image.
ConvolHighpass2	Filters an image using a 3x3 high-pass kernel and stores the absolute value (magnitude) of the result in the destination image.
ConvolKernel	Performs a convolution in image space, i.e. applies a convolution kernel.

[ConvolLaplacian4](#)

Takes the second derivative of an image using a 4-neighbor Laplacian kernel and stores the absolute value (magnitude) of the result in the destination image.

[ConvolLaplacian8](#)

Takes the second derivative of an image using an 8-neighbor Laplacian kernel and stores the absolute value (magnitude) of the result in the destination image.

[ConvolLaplacianX](#)

Takes the horizontal second derivative and stores the absolute value (magnitude) of the result in the destination image.

[ConvolLaplacianY](#)

Takes the vertical second derivative and stores the absolute value (magnitude) of the result in the destination image.

[ConvolLowpass1](#)

Filters an image using a 3x3 low-pass kernel and stores the absolute value (magnitude) of the result in the destination image.

[ConvolLowpass2](#)

Filters an image using a 3x3 low-pass kernel and stores the absolute value (magnitude) of the result in the destination image.

[ConvolLowpass3](#)

Filters an image using a 3x3 low-pass kernel.

[**ConvolPrewitt**](#)

Extracts the edges of an image by summing the absolute values of the Prewitt X and Prewitt Y derivatives and stores the absolute value (magnitude) of the result in the destination image.

[**ConvolPrewittX**](#)

Derives an image along X using a Prewitt kernel and stores the absolute value (magnitude) of the result in the destination image.

[**ConvolPrewittY**](#)

Derives an image along Y using a Prewitt kernel and stores the absolute value (magnitude) of the result in the destination image.

[**ConvolRoberts**](#)

The Roberts edge extraction filter is based on a 2x2 kernel.

[**ConvolSobel**](#)

Extracts the edges of an image by summing the absolute values of the Sobel X and Sobel Y derivatives.

[**ConvolSobelX**](#)

Derives an image along X using a Sobel kernel and stores the absolute value (magnitude) of the result in the destination image.

[**ConvolSobelY**](#)

Derives an image along Y using a Sobel kernel and stores the absolute value (magnitude) of the result in the destination image.

[ConvolSymmetricKernel](#)

[ConvolUniform](#)

Performs a convolution in image space, i.e. applies a square symmetric convolution kernel of size 3x3, 5x5 or 7x7.

[Copy](#)

Copies a source image or a constant in a destination image.

[CumulateHistogram](#)

Cumulates histogram values in another histogram.

[DilateBox](#)

Performs a dilation on an image (maximum of the gray values in a defined neighborhood) on a rectangular kernel.

[DilateDisk](#)

Performs a dilation on an image (maximum of the gray values in a defined neighborhood) on a quasi-circular kernel.

[Distance](#)

Computes the morphological distance function on a binary image (0 for black, non 0 for white).

[DoubleThreshold](#)

Converts an image by setting all pixels below the low threshold to a low value, all pixels above the high threshold to a high value, and the remaining pixels to an intermediate value.

[Equalize](#)

Equalizes an image histogram (the gray levels are re-mapped so that the histogram becomes as close to uniform as possible).

[ErodeBox](#)

Performs an erosion on an image (minimum of the gray values in a defined neighborhood) on a rectangular kernel.

[ErodeDisk](#)

Performs an erosion on an image (minimum of the gray values in a defined neighborhood) on a quasi-circular kernel.

[Focusing](#)

Returns a measure of the focusing of an image by computing the total gradient energy.

[Gain](#)

Transforms an image, applying a gain and offset to all pixels.

[GainOffset](#)

Transforms an image, applying a gain and offset to all pixels.

[GetFrame](#)

Extracts the frame of given parity from an image.

[GetProfilePeaks](#)

Detects peaks in a gray-level profile. Maxima as well as minima are considered.

[GradientScalar](#)

Computes the (scalar) gradient image derived from a given source image.

[GravityCenter](#)

Computes the coordinates of the gravity center of an image, i.e. the average coordinates of the pixels above (or on) the threshold.

[HDRFusion](#)

Fuses two images using HDR principles.

[Histogram](#)

Computes the histogram of an image (count of each gray-level value).

[HistogramThreshold](#)

Determines an appropriate threshold level based on the histogram contents, using an automatic threshold mode.

[HistogramThresholdBW16](#)

Determines an appropriate threshold level based on the histogram contents, using an automatic threshold mode.

[HitAndMiss](#)

Apply the morphological hit-and-miss transform to detect a particular pattern of foreground and background pixels in a BW8, BW16 or C24 image/ROI.

[HorizontalMirror](#)

Mirrors an image horizontally (the columns are swapped).

ImageToLineSegment	Copies the pixel values along a given line segment (arbitrarily oriented) to a vector.
ImagePath	Given a path described by coordinates in a path vector, copies the corresponding pixel values into the same vector.
IsodataThreshold	Computes a suitable threshold value for a histogram.
IsodataThresholdBW16	Computes a suitable threshold value for a histogram.
LinearTransform	Applies a general affine transformation.
LineSegmentToImage	Copies the pixel values from a vector or a constant to the pixels of a given line segment (arbitrarily oriented).
LocalAverage	Computes the average in a rectangular window centered on every pixel.
LocalDeviation	Computes the standard deviation in a rectangular window centered on every pixel.
Lut	Transforms the gray levels of an image, using a lookup table stored in a vector (of unsigned values).

MatchFrames	Determines the optimal shift amplitude by comparing two successive lines of the image.
Median	Applies a median filter to an image (median of the gray values in a 3x3 neighborhood).
ModulusImage	Prepares a lookup-table image for use for gradient magnitude computation.
MorphoGradientBox	Computes the morphological gradient of an image using a rectangular kernel.
MorphoGradientDisk	Computes the morphological gradient of an image using a quasi-circular kernel.
Normalize	Normalizes an image, i.e. applies a linear transform to the gray levels so that their average and standard deviation are imposed.
Offset	Transforms an image, applying a gain and offset to all pixels.
OpenBox	Performs an opening on an image (erosion followed by dilation) on a rectangular kernel.
OpenDisk	Performs an opening on an image (erosion followed by dilation) on a quasi-circular kernel.

Oper	Applies the desired arithmetic or logic pixel-wise operator between two images or constants.
Overlay	Overlays an image on the top of a color image, at a given position.
PathToImage	Given a path described by coordinates in a path vector, copies the pixel values from the path vector to the corresponding image pixels.
PixelAverage	Computes the average pixel value in a gray-level or color image.
PixelCompare	Counts the number of pixels differing between two images.
PixelCount	Counts the pixels in the three value classes separated by two thresholds.
PixelMax	Computes the maximum gray-level value in an image.
PixelMaxBW16	Computes the maximum gray-level value in an image.

PixelMaxBW8	Computes the maximum gray-level value in an image.
PixelMin	Computes the minimum gray-level value in an image.
PixelMinBW16	Computes the minimum gray-level value in an image.
PixelMinBW8	Computes the minimum gray-level value in an image.
PixelStat	Computes the minimum, maximum and average gray-level values in an image.
PixelStatBW16	Computes the minimum, maximum and average gray-level values in an image.
PixelStatBW8	Computes the minimum, maximum and average gray-level values in an image.
PixelStdDev	Computes the average gray-level or color value in an image, the standard deviation of the color components, and the correlation between the color components (in the case of color images).
PixelVariance	For a gray-level image, computes the mean and variance of the pixel values.

ProfileDerivative	Computes the first derivative of a profile extracted from a gray-level image.
ProjectOnAColumn	Projects an image horizontally onto a column.
ProjectOnARow	Projects an image vertically onto a row.
RealignFrame	Shifts one frame of the image horizontally.
RebuildFrame	Rebuilds one frame of the image by interpolation between the lines of the other frame.
RecursiveAverage	Applies stronger noise reduction to small variations and conversely.
Register	Registers an image by realigning one, two or three pivot points to reference positions.
RmsNoise	Computes the root-mean-square amplitude of noise, by comparing a given image to a reference image.
ScaleRotate	Re-scales an image by an arbitrary factor and/or rotates it by an arbitrary angle.
SetCircleWarp	Prepares suitable warp images for use with function EasyImage::Warp to un warp a circular ring-wedge shape into a straight rectangle.

SetFrame	Replaces the frame of given parity in an image.
SetRecursiveAverageLUT	Pre-compute the required non-linear transfer function for noise reduction by recursive temporal averaging.
SetupEqualize	Prepares a lookup-table for image equalization, using an image histogram.
Shrink	Resizes an image to a smaller size. Pre-filtering is applied to avoid aliasing.
SignalNoiseRatio	Computes the signal to noise ratio, in dB, by comparing a given image to a reference image.
SwapFrames	Interchanges the even and odd rows of an image.
Thick	Applies a thickening operation on an image, using a 3x3 kernel.
Thin	Applies a thinning operation on an image, using a 3x3 kernel.
ThreeLevelsMinResidueThreshold	Computes the two threshold values used to separate the pixels of an image in three classes.

[Threshold](#)

Binarize an image by setting pixels to two different possible values in a destination image, according to their value in a source image.

[TwoLevelsMinResidueThreshold](#)

Computes the threshold value used to separate the pixels of an image in two classes.

[Uniformize](#)

Shading correction is the process of transforming the gray or color component values of an image, using one or two reference images or vectors.

[VerticalMirror](#)

Mirrors an image vertically (the rows are swapped).

[Warp](#)

Transforms an image so that each pixels are moved to the locations specified in the "warp" images used as look-up tables.

[WeightedMoments](#)

Computes the zero-th, first, second, third or fourth order weighted moments on the gray-level image. The weight of a pixel is its gray-level value.

[WhiteTopHatBox](#)

Performs a top-hat filtering on an image (source image minus opened image) on a rectangular kernel.

WhiteTopHatDisk

EPerforms a top-hat filtering on an image (source image minus opened image) on a quasi-circular kernel.

a

sylimage.AdaptiveThreshold

Prefoms a locally adaptive threshold on the source image.

[VB6]

```
void AdaptiveThreshold(
    EROIBW8 src,
    EROIBW8 dst,
    EAdaptiveThresholdMethod method,
    Long halfKernelSize,
    Long constant
)
```

Parameters

src

-

dst

-

method

The thresholding mode, as defined by the enumeration [EAdaptiveThresholdMethod](#).

halfKernelSize

Half width of the kernel rounded down

constant

Constant offset applied to the threshold value. By default (argument omitted) **0**, i.e. no change.

Remarks

Kernel size is always odd.

EasylImage.AnalyseHistogram

Returns a floating-point statistical parameter extracted from a range of gray levels in an image histogram (most/least frequent value/frequency, min/max value, count, average, standard deviation).

[VB6]

```
Single AnalyseHistogram(  
    EBWHistogramVector histogram,  
    EHistogramFeature operation,  
    Long minimumIndex,  
    Long maximumIndex  
)
```

Parameters

histogram

Pointer to the histogram vector.

operation

Parameter to be computed, as defined by [EHistogramFeature](#).

minimumIndex

Starting index of the gray-level range.

maximumIndex

Ending index of the gray-level range.

EasylImage.AnalyseHistogramBW16

Returns a floating-point statistical parameter extracted from a range of gray levels in an image histogram (most/least frequent value/frequency, min/max value, count, average, standard deviation).

[VB6]

```
Single AnalyseHistogramBW16(
    EBWHistogramVector histogram,
    EHistogramFeature operation,
    Long minimumIndex,
    Long maximumIndex
)
```

Parameters*histogram*

Pointer to the histogram vector.

*operation*Parameter to be computed, as defined by [EHistogramFeature](#).*minimumIndex*

Starting index of the gray-level range.

maximumIndex

Ending index of the gray-level range.

EasylImage.Area

Counts the pixels whose values are above (or on) a threshold.

[VB6]

```
void Area(
    EROIBW8 sourceImage,
    EBW8 threshold,
    Long numberOfPixelsAboveThreshold
)

void Area(
    EROIBW16 sourceImage,
    EBW16 threshold,
    Long numberOfPixelsAboveThreshold
)
```

```

void Area(
    EROIBW8 sourceImage,
    EROIBW8 mask,
    EBW8 threshold,
    Long numberOfPixelsAboveThreshold
)

void Area(
    EROIBW16 sourceImage,
    EROIBW8 mask,
    EBW16 threshold,
    Long numberOfPixelsAboveThreshold
)

```

Parameters*sourceImage*

Pointer to the source image/ROI.

threshold

The pixel thresholding value used to count the pixels

numberOfPixelsAboveThreshold

Reference to the count of pixels above or equal to the threshold.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

EasylImage.AreaDoubleThreshold

Counts the pixels whose values are comprised between (or on) two thresholds.

[VB6]

```

void AreaDoubleThreshold(
    EROIBW8 sourceImage,
    EBW8 lowThreshold,
    EBW8 highThreshold,
    Long numberOfPixelsBetweenThresholds
)

```

```

void AreaDoubleThreshold(
    EROIBW16 sourceImage,
    EBW16 lowThreshold,
    EBW16 highThreshold,
    Long numberOfPixelsBetweenThresholds
)

void AreaDoubleThreshold(
    EROIBW8 sourceImage,
    EROIBW8 mask,
    EBW8 lowThreshold,
    EBW8 highThreshold,
    Long numberOfPixelsBetweenThresholds
)

void AreaDoubleThreshold(
    EROIBW16 sourceImage,
    EROIBW8 mask,
    EBW16 lowThreshold,
    EBW16 highThreshold,
    Long numberOfPixelsBetweenThresholds
)

```

Parameters

sourceImage

Pointer to the source image/ROI.

lowThreshold

Inferior threshold.

highThreshold

Superior threshold.

numberOfPixelsBetweenThresholds

Reference to the count of pixels that are above or equal to the inferior threshold, and strictly below the superior threshold.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

EasylImage.ArgumentsImage

Prepares a lookup-table image for use for gradient argument computation.

[VB6]

```

void ArgumentImage(
    EImageBW8 destinationImage,
    EBW8 phase,
    Single period
)

void ArgumentImage(
    EImageBW8 destinationImage
)

void ArgumentImage(
    EImageBW8 destinationImage,
    EBW8 phase
)

```

Parameters

destinationImage

Pointer to the destination image.

phase

Argument value corresponding to the horizontal direction, in 256-th (65,536-th) of the period (by default, **phase = 0**).

period

Range of argument values corresponding to the **0..255 (0..65535)** interval, in the current angle unit (by default, **period = 0**).

Remarks

The scale and phase of the gradient argument can be adjusted. The argument angles are counted clockwise on a **0..255** scale in the **BW8** context and on a **0..65535** scale in the **BW16** one, corresponding to a specified range (full turn by default, specified period otherwise). The argument phase is counted on a **0..255** scale or on a **0..65535** scale too. Angle values outside the **0..255 (0..65535)** interval are wrapped. The period length is given in the current angle unit. [ArgumentImage](#) sets a lookup-table image for use with function [EasylImage::GradientScalar](#), ready to compute the argument of the gradient in the source image, i.e. its direction. The argument will be returned as a value in range **0..255** suitable for storage in an [EImageBW8](#) or as a value in the range **0..65535** suitable for storage in an [EImageBW16](#). The phase of the argument can be adjusted.

EasylImage.AutoThreshold

Returns a suitable threshold value for a gray-level image binarization.

[VB6]

```

EBW8 AutoThreshold(
    EROIBW8 sourceImage,
    EThresholdMode thresholdMode,
    Single relativeThresholdMode
)

EBW16 AutoThreshold(
    EROIBW16 sourceImage,
    EThresholdMode thresholdMode,
    Single relativeThresholdMode
)

EBW8 AutoThreshold(
    EROIBW8 sourceImage,
    EROIBW8 mask,
    EThresholdMode thresholdMode,
    Single relativeThresholdMode
)

EBW16 AutoThreshold(
    EROIBW16 sourceImage,
    EROIBW8 mask,
    EThresholdMode thresholdMode,
    Single relativeThresholdMode
)

```

Parameters

sourceImage

Pointer to the source image/ROI.

thresholdMode

The thresholding mode, as defined by the enumeration [EThresholdMode](#). To use absolute thresholding, use directly the threshold value instead.

relativeThresholdMode

Fraction of the image pixels that will be set below the threshold. Only used when the threshold value is [EThresholdMode_Relative](#) (by default, **relativeThresholdMode = 0.5**).

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

Remarks

Several modes are available: absolute (the threshold value is given readily in the **thresholdMode** parameter), relative (the threshold value is computed to obtain a desired fraction of the image pixels) or automatic (using three different criteria).

It is possible that, in the automatic or relative thresholding modes, the computed threshold exceeds the dynamic range of the return type. In this case, the value is clipped to the maximum value that is representable in the return type.

EasylImage.BiLevelBlackTopHatBox

Performs a top-hat filtering on a bilevel image (closed image minus source image) on a rectangular kernel.

[VB6]

```
void BiLevelBlackTopHatBox(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as the source image.

halfOfKernelWidth

Half of the box width minus one (by default, **halfOfKernelWidth = 1**; **0** is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as **halfOfKernelWidth**; **0** is allowed).

Remarks

This filter enhances the thin black features.

EasylImage.BiLevelBlackTopHatDisk

Performs a top-hat filtering on a bilevel image (closed image minus source image) on a quasi-circular kernel.

[VB6]

```
void BiLevelBlackTopHatDisk(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as the source image.

halfOfKernelWidth

Half width of the kernel minus one (by default, **halfOfKernelWidth = 1; 0** is allowed).

Remarks

This filter enhances the thin black features.

EasylImage.BiLevelCloseBox

Performs a closing on a bilevel image (dilation followed by erosion) on a rectangular kernel.

[VB6]

```
void BiLevelCloseBox(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half of the box width minus one (by default, **halfOfKernelWidth = 1; 0** is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as **halfOfKernelWidth; 0** is allowed).

EasylImage.BiLevelCloseDisk

Performs a closing on a bilevel image (dilation followed by erosion) on a quasi-circular kernel.

[VB6]

```
void BiLevelCloseDisk(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half width of the kernel minus one (by default, **halfOfKernelWidth = 1; 0** is allowed).

EasylImage.BiLevelDilateBox

Performs a dilation on a bilevel image (maximum of the pixel values in a defined neighborhood) on a rectangular kernel. For bilevel images, this maximum can either be 0 (if all pixels are black in the given neighborhood), or the maximum possible pixel value.

[VB6]

```
void BiLevelDilateBox(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half of the box width minus one (by default, **halfOfKernelWidth = 1; 0** is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as **halfOfKernelWidth**; **0** is allowed).

EasylImage.BiLevelDilateDisk

Performs a dilation on a bilevel image (maximum of the pixel values in a defined neighborhood) on a quasi-circular kernel. For bilevel images, this maximum can either be 0 (if all pixels are black in the given neighborhood), or the maximum possible pixel value.

[VB6]

```
void BiLevelDilateDisk(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half width of the kernel minus one (by default, **halfOfKernelWidth =1; 0** is allowed).

EasylImage.BiLevelErodeBox

Performs an erosion on a bilevel image (minimum of the pixel values in a defined neighborhood) on a rectangular kernel. For bilevel images, this minimum can either be 0, or the maximum possible pixel value (if all pixels are white in the given neighborhood)

[VB6]

```
void BiLevelErodeBox(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half of the box width minus one (by default, **halfOfKernelWidth = 1**; **0** is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as **halfOfKernelWidth**; **0** is allowed).

EasylImage.BiLevelErodeDisk

Performs an erosion on a bilevel image (minimum of the pixel values in a defined neighborhood) on a quasi-circular kernel. For bilevel images, this minimum can either be 0, or the maximum possible pixel value (if all pixels are white in the given neighborhood)

[VB6]

```
void BiLevelErodeDisk(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half width of the kernel minus one (by default, **halfOfKernelWidth = 1**; **0** is allowed).

EasylImage.BiLevelMedian

Applies a median filter to a bilevel image (median of the gray values in a 3x3 neighborhood).

[VB6]

```
void BiLevelMedian(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as source image.

EasylImage.BiLevelMorphoGradientBox

Computes the morphological gradient of a bilevel image using a rectangular kernel.

[VB6]

```
void BiLevelMorphoGradientBox(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

*destinationImage*Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).*halfOfKernelWidth*Half of the box width minus one (by default, **halfOfKernelWidth = 1**; **0** is allowed).*halfOfKernelHeight*Half of the box height minus one (by default, same as **halfOfKernelWidth**; **0** is allowed).

Remarks

The morphological gradient is the difference between the dilation and the erosion of the image, using the same structuring element. The kernel size is a pair of odd numbers; they must be halved before they are passed. For instance, a 3x5 kernel is passed as 1x2.

EasylImage.BiLevelMorphoGradientDisk

Computes the morphological gradient of a bilevel image using a quasi-circular kernel.

[VB6]

```
void BiLevelMorphoGradientDisk(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half width of the kernel minus one (by default, **halfOfKernelWidth = 1**; **0** is allowed).

Remarks

The morphological gradient is the difference between the dilation and the erosion of the image, using the same structuring element.

EasylImage.BiLevelOpenBox

Performs an opening on a bilevel image (erosion followed by dilation) on a rectangular kernel.

[VB6]

```
void BiLevelOpenBox(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

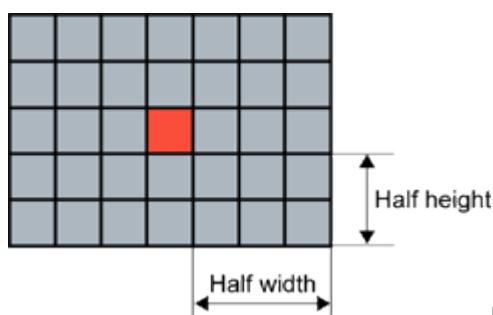
halfOfKernelWidth

Half of the box width minus one, as shown on the picture below (by default, **halfOfKernelWidth =1; 0** is allowed).

halfOfKernelHeight

Half of the box height minus one, as shown on the picture below (by default, same as **halfOfKernelWidth; 0** is allowed).

Remarks



Rectangular kernel of half width = 3 and half height = 2

EasylImage.BiLevelOpenDisk

Performs an opening on a bilevel image (erosion followed by dilation) on a quasi-circular kernel.

[VB6]

```
void BiLevelOpenDisk(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half width of the kernel minus one, as shown on the picture below (by default, **halfOfKernelWidth = 1**; **0** is allowed).

EasylImage.BiLevelThick

Applies a thickening operation on a bilevel image, using a 3x3 kernel.

[VB6]

```
void BiLevelThick(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    EKernel thickeningKernel,
    EKernelRotation rotationMode,
    Long numberofIterations
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as source image.

thickeningKernel

Pointer to the thickening kernel.

rotationMode

Rotation mode, as defined by [EKernelRotation](#).

numberOfIterations

Number of iterations to apply. **0** indicates stop when convergence is reached. Upon return, gives the number of passes actually performed. If the rotation mode is set to either [EKernelRotation_Clockwise](#) or [EKernelRotation_Anticlockwise](#), a pass comprises eight kernel rotations.

Remarks

The thickening kernel coefficients must be **0** (matching black pixel, value 0), **1** (matching non black pixel, value > 0) or **-1** (don't care). When a match is found between the kernel coefficients and the neighborhood of a pixel, the pixel value is set to **255**.

EasylImage.BiLevelThin

Applies a thinning operation on a bilevel image, using a 3x3 kernel.

[VB6]

```
void BiLevelThin(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    EKernel thinningKernel,
    EKernelRotation rotationMode,
    Long numberOfIterations
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as source image.

thinningKernel

Pointer to the thinning kernel.

rotationMode

Rotation mode, as defined by [EKernelRotation](#).

numberOfIterations

Number of iterations to apply. **0** indicates stop when convergence is reached. Upon return, gives the number of passes actually performed. If the rotation mode is set to either [EKernelRotation_Clockwise](#) or [EKernelRotation_Anticlockwise](#), a pass comprises eight kernel rotations.

Remarks

The thinning kernel coefficients must be **0** (matching black pixel, value 0), **1** (matching non black pixel, value > 0) or **-1** (don't care). When a match is found between the kernel coefficients and the neighborhood of a pixel, the pixel value is set to 0.

EasylImage.BiLevelWhiteTopHatBox

Performs a top-hat filtering on a bilevel image (source image minus opened image) on a rectangular kernel.

[VB6]

```
void BiLevelWhiteTopHatBox(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as the source image.

halfOfKernelWidth

Half of the box width minus one (by default, **halfOfKernelWidth = 1**; **0** is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as **halfOfKernelWidth**; **0** is allowed).

Remarks

This filter enhances the thin white features.

EasylImage.BiLevelWhiteTopHatDisk

Performs a top-hat filtering on a bilevel image (source image minus opened image) on a quasi-circular kernel.

[VB6]

```
void BiLevelWhiteTopHatDisk(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as the source image.

halfOfKernelWidth

Half width of the kernel minus one (by default, **halfOfKernelWidth = 1; 0** is allowed).

Remarks

This filter enhances the thin white features.

EasylImage.BinaryMoments

Computes the zero-th, first or second order moments on the binarized image, i.e. with a unit weight for those pixels with a value above or equal to the threshold, and zero otherwise.

[VB6]

```
void BinaryMoments(
    EROIBW8 sourceImage,
    Long threshold,
    Single M,
    Single Mx,
    Single My
)

void BinaryMoments(
    EROIBW16 sourceImage,
    Long threshold,
    Single M,
    Single Mx,
    Single My
)

void BinaryMoments(
    EROIBW8 sourceImage,
    EROIBW8 mask,
    Long threshold,
    Single M,
    Single Mx,
    Single My
)

void BinaryMoments(
    EROIBW16 sourceImage,
    EROIBW8 mask,
    Long threshold,
    Single M,
    Single Mx,
    Single My
)

void BinaryMoments(
    EROIBW8 sourceImage,
    Long threshold,
    Single M,
    Single Mx,
    Single My,
    Single Mxx,
    Single Mxy,
    Single Myy
)
```

```
void BinaryMoments(
    EROIBW16 sourceImage,
    Long threshold,
    Single M,
    Single Mx,
    Single My,
    Single Mxx,
    Single Mxy,
    Single Myy
)

void BinaryMoments(
    EROIBW8 sourceImage,
    EROIBW8 mask,
    Long threshold,
    Single M,
    Single Mx,
    Single My,
    Single Mxx,
    Single Mxy,
    Single Myy
)

void BinaryMoments(
    EROIBW16 sourceImage,
    EROIBW8 mask,
    Long threshold,
    Single M,
    Single Mx,
    Single My,
    Single Mxx,
    Single Mxy,
    Single Myy
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

threshold

Binarization threshold.

M

Reference to the zero-th order moment (area).

Mx

Reference to the first-order, uncentered moments (weighted sum of abscissas).

My

Reference to the first-order, uncentered moments (weighted sum of ordinates).

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

Mxx

Reference to the second-order, uncentered moments (weighted sum of squared abscissas).

Mxy

Reference to the second-order, uncentered moments (weighted sum of cross-product of abscissas and ordinates).

Myy

Reference to the second-order, uncentered moments (weighted sum of squared ordinates).

EasylImage.BlackTopHatBox

Performs a top-hat filtering on an image (closed image minus source image) on a rectangular kernel.

[VB6]

```
void BlackTopHatBox(
    EROIBW1 sourceImage,
    EROIBW1 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

void BlackTopHatBox(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

void BlackTopHatBox(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)
```

```
void BlackTopHatBox(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as the source image.

halfOfKernelWidth

Half of the box width minus one (by default, **halfOfKernelWidth = 1**; **0** is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as **halfOfKernelWidth**; **0** is allowed).

Remarks

This filter enhances the thin black features.

Easylimage.BlackTopHatDisk

Performs a top-hat filtering on an image (closed image minus source image) on a quasi-circular kernel.

[VB6]

```
void BlackTopHatDisk(
    EROIBW1 sourceImage,
    EROIBW1 destinationImage,
    Long halfOfKernelWidth
)

void BlackTopHatDisk(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth
)
```

```

void BlackTopHatDisk(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Long halfOfKernelWidth
)

void BlackTopHatDisk(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Long halfOfKernelWidth
)

```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as the source image.

halfOfKernelWidth

Half width of the kernel minus one (by default, **halfOfKernelWidth = 1; 0** is allowed).

Remarks

This filter enhances the thin black features.

Easylimage.CloseBox

Performs a closing on an image (dilation followed by erosion) on a rectangular kernel.

[VB6]

```

void CloseBox(
    EROIBW1 sourceImage,
    EROIBW1 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

```

```

void CloseBox(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

void CloseBox(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

void CloseBox(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half of the box width minus one (by default, **halfOfKernelWidth = 1**; **0** is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as **halfOfKernelWidth**; **0** is allowed).

EasylImage.CloseDisk

Performs a closing on an image (dilation followed by erosion) on a quasi-circular kernel.

[VB6]

```

void CloseDisk(
    EROIBW1 sourceImage,
    EROIBW1 destinationImage,
    Long halfOfKernelWidth
)

void CloseDisk(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth
)

void CloseDisk(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Long halfOfKernelWidth
)

void CloseDisk(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Long halfOfKernelWidth
)

```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half width of the kernel minus one (by default, **halfOfKernelWidth = 1; 0** is allowed).

Easylimage.Contour

Follows the *contour* of an object.

[VB6]

```
void Contour(
    EROIBW8 sourceImage,
    EContourMode contourMode,
    Long startX,
    Long startY,
    EContourThreshold thresholdMode,
    Long threshold,
    EConnexity connexity,
    EPathVector path
)

void Contour(
    EROIBW16 sourceImage,
    EContourMode contourMode,
    Long startX,
    Long startY,
    EContourThreshold thresholdMode,
    Long threshold,
    EConnexity connexity,
    EPathVector path
)

void Contour(
    EROIBW8 sourceImage,
    EContourMode contourMode,
    Long startX,
    Long startY,
    EContourThreshold thresholdMode,
    Long threshold,
    EConnexity connexity,
    EBW8PathVector path,
    Boolean freeman
)

void Contour(
    EROIBW16 sourceImage,
    EContourMode contourMode,
    Long startX,
    Long startY,
    EContourThreshold thresholdMode,
    Long threshold,
    EConnexity connexity,
    EBW16PathVector path,
    Boolean freeman
)
```

Parameters*sourceImage*

Pointer to the source image/ROI.

*contourMode*Traversal mode, as defined by [EContourMode](#).*startX*

Start point abscissa.

startY

Start point ordinate.

*thresholdMode*Thresholding mode as defined by [EThresholdMode](#).*threshold*

Threshold level.

*connexity*Contour connexity, as defined by [EConnexity](#).*path*

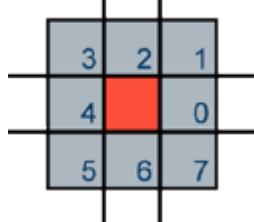
Pointer to the destination vector.

freeman

Specifies if Freeman codes are to be retrieved rather than pixel values.

Remarks

A threshold is applied so that objects become blobs. The contour is a closed or not (see property [Get/SetClosed](#)) connected path, forming the boundary of the blob. When destination vector is an [EBW8PathVector](#) or a [EBW16PathVector](#), this vector can contain two different information. If the **bFreeman** argument is **FALSE**, which is the default value, member **m_bw8** (**16**)**Pixel** in the vector elements contains the gray-level value of the contour pixels. If it is **TRUE**, the member instead gives the Freeman code leading from a pixel to next. The Freeman codes are numbered from 0 in the horizontal direction and incremented anticlockwise.



Freeman code, leading from a pixel to another adjacent pixel

Easylimage.Convert

Transforms the contents of an image to an image of another type.

[VB6]

```
void Convert(
    EROIC24 sourceImage,
    EROIBW8 destinationImage
)

void Convert(
    EROIBW8 sourceImage,
    EROIC24 destinationImage
)

void Convert(
    EROIC24 sourceImage,
    EROIC15 destinationImage
)

void Convert(
    EROIC15 sourceImage,
    EROIC24 destinationImage
)

void Convert(
    EROIBW8 sourceImage,
    EROIC15 destinationImage
)

void Convert(
    EROIC15 sourceImage,
    EROIBW8 destinationImage
)

void Convert(
    EROIC24 sourceImage,
    EROIC16 destinationImage
)

void Convert(
    EROIC16 sourceImage,
    EROIC24 destinationImage
)

void Convert(
    EROIBW8 sourceImage,
    EROIC16 destinationImage
)
```

```
void Convert(
    EROIC16 sourceImage,
    EROIBW8 destinationImage
)

void Convert(
    EROIC24 sourceImage,
    EROIC24A destinationImage
)

void Convert(
    EROIC24A sourceImage,
    EROIC24 destinationImage
)

void Convert(
    EROIBW8 sourceImage,
    EROIBW1 destinationImage
)

void Convert(
    EROIBW16 sourceImage,
    EROIBW1 destinationImage
)

void Convert(
    EROIBW32 sourceImage,
    EROIBW1 destinationImage
)

void Convert(
    EROIBW32 sourceImage,
    EROIBW16 destinationImage,
    Long rightShift
)

void Convert(
    EROIBW32 sourceImage,
    EROIBW8 destinationImage,
    Long rightShift
)

void Convert(
    EROIBW16 sourceImage,
    EROIBW8 destinationImage,
    Long rightShift
)
```

```
void Convert(
    EROIBW8 sourceImage,
    EROIBW16 destinationImage,
    Long leftShift
)

void Convert(
    EROIBW8 sourceImage,
    EROIBW32 destinationImage,
    Long leftShift
)

void Convert(
    EROIBW16 sourceImage,
    EROIBW32 destinationImage,
    Long leftShift
)

void Convert(
    EROIC24 sourceImage,
    EROIBW8 sourceImageAlpha,
    EROIC24A destinationImage
)

void Convert(
    EROIC24A sourceImage,
    EROIC24 destinationImage,
    EROIBW8 destinationImageAlpha
)

void Convert(
    EROIBW1 sourceImage,
    EROIBW8 destinationImage,
    EBW8 highValue
)

void Convert(
    EROIBW1 sourceImage,
    EROIBW8 destinationImage
)

void Convert(
    EROIBW1 sourceImage,
    EROIBW16 destinationImage,
    EBW16 highValue
)
```

```
void Convert(
    EROIBW1 sourceImage,
    EROIBW16 destinationImage
)

void Convert(
    EROIBW1 sourceImage,
    EROIBW32 destinationImage,
    EBW32 highValue
)

void Convert(
    EROIBW1 sourceImage,
    EROIBW32 destinationImage
)

void Convert(
    EROIC48 sourceImage,
    EROIC24 destinationImage,
    Long rightShift
)

void Convert(
    EROIC24 sourceImage,
    EROIC48 destinationImage,
    Long leftShift
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

rightShift

Right shift amplitude. By default, left justified data is assumed.

leftShift

Left shift amplitude. By default, left justified data is assumed.

sourceImageAlpha

Pointer to the source alpha component ([ElImageBW8/EROIBW8](#)).

destinationImageAlpha

Pointer to the destination alpha component ([ElImageBW8/EROIBW8](#)).

highValue

In the case of black and white source images/ROIs, indicates to which gray level the value **1** should be mapped. By default, **1** is mapped to the highest allowed value for the destination image/ROI.

Remarks

Conversion to a black and white image (BW1)Turns an 8-bit gray-level image into a black and white image. Turns a 16-bit gray-level image into a black and white image. Turns a 32-bit gray-level image into a black and white image. Source pixels whose values is 0 are converted to black. All other source pixel values are converted to white. Conversion to a 8-bit gray-level image (BW8)Turns a black and white image into an 8-bit gray-level image. Turns a 16-bit gray-level image into an 8-bit gray-level image. A right shift can be applied to the 16-bit data to adjust the magnitude, depending on how the 16-bit data is organized. For instance, if the source image holds 10 significant bits right justified, a right shift of 2 is required to drop the 2 low order bits; if the source image holds 12 bits left justified, a right shift of 8 is required and the 4 low order bits will be truncated. Turns an [EC15](#), [EC16](#) or [EC24](#) color image into an [EBW8](#) gray-level image. The 3 color components are simply averaged, giving the intensity component of the ISH color system. Conversion to a 16-bit gray-level image (BW16)Turns a black and white image into a 16-bit gray-level image. Turns an 8-bit gray-level image into a 16-bit gray-level image. A left shift can be applied to the 8-bit data to adjust the magnitude, depending on how the 16-bit data is organized. For instance, if the destination image holds 10 significant bits right justified, a shift of 2 is required; if the destination image holds 12 bits left justified, a shift of 8 is required. Conversion to a 32-bit gray-level image (BW32)Turns a black and white image into a 32-bit gray-level image. Conversion to color imagesTurns an 8-bit gray-level image into a true color equivalent. The color components are all set equal to the corresponding gray-level value. Converts between standard and Windows' packing RGB color formats. When converting from an [EC24](#) image to a [EC15](#) or [EC16](#) one, only the 5 (or 6) most significant bits of each color component are retained. Converts between RGB 24-bit color image and RGB32 (also known as RGBA) color image. When converting from [EC24](#) to [EC24A](#), you can choose to provide or not the alpha component. On the other hand, when converting from [EC24A](#) to [EC24](#), you can choose to conserve or not the alpha component. The alpha component is retrieved and set using an [EImageBW8/EROIBW8](#).

EasylImage.ConvertTo422

Turns an 8-bit gray-level image into a YUV 4:2:2 encoded color image.

[VB6]

```
void ConvertTo422(
    EROIBW8 sourceImage,
    EROIBW16 destinationImage
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

Remarks

The Y component is set to the corresponding gray-level values, while the U and V components are set to **128** (achromatic light).

EasylImage.ConvolveGaussian

Applies Gaussian filtering (binomial weights) in rectangular kernel of odd size.

[VB6]

```
void ConvolveGaussian(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

void ConvolveGaussian(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)
```

```

void ConvolGaussian(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

void ConvolGaussian(
    EBW8Vector sourceImage,
    EBW8Vector destinationImage,
    Long halfOfKernelWidth
)

void ConvolGaussian(
    EBW16Vector sourceImage,
    EBW16Vector destinationImage,
    Long halfOfKernelWidth
)

```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half of the box width minus one (by default, **halfOfKernelWidth = 1**; **0** is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as **halfOfKernelHeight**; **0** is allowed).

EasylImage.ConvolGradient

Extracts the edges of an image by summing the absolute values of the Gradient X and Gradient Y derivatives and stores the absolute value (magnitude) of the result in the destination image.

[VB6]

```

void ConvolGradient(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

void ConvolGradient(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)

void ConvolGradient(
    EROIC24 sourceImage,
    EROIC24 destinationImage
)

```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

Easylimage.ConvolGradientX

Derives an image along X using a Gradient kernel and stores the absolute value (magnitude) of the result in the destination image.

[VB6]

```

void ConvolGradientX(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

void ConvolGradientX(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)

```

```
void ConvolGradientX(
    EROIC24 sourceImage,
    EROIC24 destinationImage
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

Remarks

Filtering kernel: 0 0 0 -1 0 1 0 0 0

EasylImage.ConvolGradientY

Derives an image along Y using a Gradient kernel and stores the absolute value (magnitude) of the result in the destination image.

[VB6]

```
void ConvolGradientY(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

void ConvolGradientY(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)

void ConvolGradientY(
    EROIC24 sourceImage,
    EROIC24 destinationImage
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

Remarks

Filtering kernel: 0 -1 0 0 0 0 0 1 0

EasylImage.ConvolveHighpass1

Filters an image using a 3x3 high-pass kernel and stores the absolute value (magnitude) of the result in the destination image.

[VB6]

```
void ConvolveHighpass1(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

void ConvolveHighpass1(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)

void ConvolveHighpass1(
    EROIC24 sourceImage,
    EROIC24 destinationImage
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

Remarks

Filtering kernel: 0 -1 0 1 5 -1 0 -1 0

EasylImage.ConvolveHighpass2

Filters an image using a 3x3 high-pass kernel and stores the absolute value (magnitude) of the result in the destination image.

[VB6]

```
void ConvolveHighpass2(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

void ConvolveHighpass2(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)

void ConvolveHighpass2(
    EROIC24 sourceImage,
    EROIC24 destinationImage
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

Remarks

Filtering kernel: -1 -1 -1-1 9 -1-1 -1 -1

EasylImage.ConvolveKernel

Performs a convolution in image space, i.e. applies a convolution kernel.

```
[VB6]

void ConvolKernel(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    EKernel kernel
)

void ConvolKernel(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    EKernel kernel
)

void ConvolKernel(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    EKernel kernel
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

kernel

Pointer to the convolution kernel.

Remarks

Please note that **sourcelImage** and **destinationImage** must be different objects.

EasylImage.ConvolLaplacian4

Takes the second derivative of an image using a 4-neighbor Laplacian kernel and stores the absolute value (magnitude) of the result in the destination image.

```
[VB6]
```

```

void ConvolLaplacian4(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

void ConvolLaplacian4(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)

void ConvolLaplacian4(
    EROIC24 sourceImage,
    EROIC24 destinationImage
)

```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

Remarks

Filtering kernel: 0 1 01 -4 10 1 0

EasylImage.ConvolLaplacian8

Takes the second derivative of an image using an 8-neighbor Laplacian kernel and stores the absolute value (magnitude) of the result in the destination image.

[VB6]

```

void ConvolLaplacian8(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

```

```

void ConvolveLaplacian8(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)

void ConvolveLaplacian8(
    EROIC24 sourceImage,
    EROIC24 destinationImage
)

```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

Remarks

Filtering kernel: 1 1 11 -8 11 1 1

EasylImage.ConvolveLaplacianX

Takes the horizontal second derivative and stores the absolute value (magnitude) of the result in the destination image.

[VB6]

```

void ConvolveLaplacianX(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

void ConvolveLaplacianX(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)

void ConvolveLaplacianX(
    EROIC24 sourceImage,
    EROIC24 destinationImage
)

```

Parameters*sourceImage*

Pointer to the source image/ROI.

*destinationImage*Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).**Remarks**

Filtering kernel: 1 -2 1

Easylimage.ConvolveLaplacianY

Takes the vertical second derivative and stores the absolute value (magnitude) of the result in the destination image.

[VB6]

```
void ConvolveLaplacianY(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

void ConvolveLaplacianY(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)

void ConvolveLaplacianY(
    EROIC24 sourceImage,
    EROIC24 destinationImage
)
```

Parameters*sourceImage*

Pointer to the source image/ROI.

*destinationImage*Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).**Remarks**

Filtering kernel: 1-2 1

EasylImage.ConvolveLowpass1

Filters an image using a 3x3 low-pass kernel and stores the absolute value (magnitude) of the result in the destination image.

[VB6]

```
void ConvolveLowpass1(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

void ConvolveLowpass1(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)

void ConvolveLowpass1(
    EROIC24 sourceImage,
    EROIC24 destinationImage
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

Remarks

Filtering kernel: 1 1 1 1 1 1 1 1

EasylImage.ConvolveLowpass2

Filters an image using a 3x3 low-pass kernel and stores the absolute value (magnitude) of the result in the destination image.

[VB6]

```

void ConvolveLowpass2(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

void ConvolveLowpass2(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)

void ConvolveLowpass2(
    EROIC24 sourceImage,
    EROIC24 destinationImage
)

```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

Remarks

Filtering kernel: 1 1 1 1 0 1 1 1 1

EasylImage.ConvolveLowpass3

Filters an image using a 3x3 low-pass kernel.

[VB6]

```

void ConvolveLowpass3(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

```

```

void ConvolLowpass3(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)

void ConvolLowpass3(
    EROIC24 sourceImage,
    EROIC24 destinationImage
)

```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

Remarks

Filtering kernel: 1 2 12 4 21 2 1

EasylImage.ConvolPrewitt

Extracts the edges of an image by summing the absolute values of the Prewitt X and Prewitt Y derivatives and stores the absolute value (magnitude) of the result in the destination image.

[VB6]

```

void ConvolPrewitt(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

void ConvolPrewitt(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)

```

```
void ConvolPrewitt(
    EROIC24 sourceImage,
    EROIC24 destinationImage
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

Easylimage.ConvolPrewittX

Derives an image along X using a Prewitt kernel and stores the absolute value (magnitude) of the result in the destination image.

[VB6]

```
void ConvolPrewittX(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

void ConvolPrewittX(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)

void ConvolPrewittX(
    EROIC24 sourceImage,
    EROIC24 destinationImage
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

Remarks

Filtering kernel: -1 0 1-1 0 1-1 0 1

EasylImage.ConvolvePrewittY

Derives an image along Y using a Prewitt kernel and stores the absolute value (magnitude) of the result in the destination image.

```
[VB6]
void ConvolvePrewittY(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

void ConvolvePrewittY(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)

void ConvolvePrewittY(
    EROIC24 sourceImage,
    EROIC24 destinationImage
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

Remarks

Filtering kernel: -1 -1 -1 0 0 1 1 1

EasylImage.ConvolveRoberts

The Roberts edge extraction filter is based on a 2x2 kernel.

```
[VB6]

void ConvolveRoberts(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

void ConvolveRoberts(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)

void ConvolveRoberts(
    EROIC24 sourceImage,
    EROIC24 destinationImage
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

Remarks

It computes the sum of absolute differences of the pixel values in the diagonal directions.

EasylImage.ConvolveSobel

Extracts the edges of an image by summing the absolute values of the Sobel X and Sobel Y derivatives.

```
[VB6]

void ConvolSobel(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

void ConvolSobel(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)

void ConvolSobel(
    EROIC24 sourceImage,
    EROIC24 destinationImage
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

EasylImage.ConvolSobelX

Derives an image along X using a Sobel kernel and stores the absolute value (magnitude) of the result in the destination image.

```
[VB6]

void ConvolSobelX(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

void ConvolSobelX(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)
```

```
void ConvolSobelX(
    EROIC24 sourceImage,
    EROIC24 destinationImage
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

Remarks

Filtering kernel: -1 0 1-2 0 2-1 0 1

EasylImage.ConvolSobelY

Derives an image along Y using a Sobel kernel and stores the absolute value (magnitude) of the result in the destination image.

[VB6]

```
void ConvolSobelY(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

void ConvolSobelY(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)

void ConvolSobelY(
    EROIC24 sourceImage,
    EROIC24 destinationImage
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

Remarks

Filtering kernel: -1 -2 -1 0 0 0 1 2 1

EasylImage.ConvolveSymmetricKernel

Performs a convolution in image space, i.e. applies a square symmetric convolution kernel of size 3x3, 5x5 or 7x7.

[VB6]

```
void ConvolveSymmetricKernel(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    EKernel kernel
)

void ConvolveSymmetricKernel(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    EKernel kernel
)

void ConvolveSymmetricKernel(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    EKernel kernel
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

kernel

Pointer to the convolution kernel.

Remarks

This function is a synonym for [EasylImage::ConvolveKernel](#).

EasylImage.ConvolUniform

Applies strong low-pass filtering to an image by using a uniform rectangular kernel of odd size.

[VB6]

```
void ConvolUniform(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

void ConvolUniform(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

void ConvolUniform(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

void ConvolUniform(
    EBW8Vector sourceVector,
    EBW8Vector destinationVector,
    Long halfOfKernelWidth
)

void ConvolUniform(
    EBW16Vector sourceVector,
    EBW16Vector destinationVector,
    Long halfOfKernelWidth
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image) and the default value for **un32HalfWidth** **(1)** has to be used.

halfOfKernelWidth

Half of the box width minus one (by default, **halfOfKernelWidth = 1; 0** is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as **halfOfKernelWidth; 0** is allowed).

sourceVector

Pointer to the source vector.

destinationVector

Pointer to the destination vector. If **NULL** (default), this operation is destructive (i.e. applied to the source vector) and the default value for **un32HalfWidth** **(1)** has to be used.

Remarks

This filter replaces every pixel values by the arithmetic mean of the neighboring values in a rectangular window. To handle pixels along edges, the source pixels are replicated outwards as many times as required. A very nice feature of this function is that its running time does not depend on the kernel size!

EasylImage.Copy

Copies a source image or a constant in a destination image.

[VB6]

```
void Copy(
    EROIBW1 sourceImage,
    EROIBW1 destinationImage
)

void Copy(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)

void Copy(
    EROIBW32 sourceImage,
    EROIBW32 destinationImage
)
```

```
void Copy(
    EROIC24 sourceImage,
    EROIC24 destinationImage
)

void Copy(
    EROIC15 sourceImage,
    EROIC15 destinationImage
)

void Copy(
    EROIC16 sourceImage,
    EROIC16 destinationImage
)

void Copy(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

void Copy(
    EROIC48 sourceImage,
    EROIC48 destinationImage
)

void Copy(
    EBW1 constant,
    EROIBW1 destinationImage
)

void Copy(
    EBW16 constant,
    EROIBW16 destinationImage
)

void Copy(
    EBW32 constant,
    EROIBW32 destinationImage
)

void Copy(
    EC24 constant,
    EROIC24 destinationImage
)

void Copy(
    EC15 constant,
    EROIC15 destinationImage
)
```

```
void Copy(
    EC16 constant,
    EROIC16 destinationImage
)

void Copy(
    EBW8 constant,
    EROIBW8 destinationImage
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

constant

Gray-level or color constant.

EasylImage.CumulateHistogram

Cumulates histogram values in another histogram.

[VB6]

```
void CumulateHistogram(
    EBWHistogramVector sourceVector,
    EBWHistogramVector destinationVector
)
```

Parameters

sourceVector

Pointer to the source vector.

destinationVector

Pointer to the destination vector.

Remarks

Calling this function after [EasylImage::Histogram](#) allows you to compute the cumulative histogram of an image, i.e. the count of pixels below a given threshold value (instead of the count of pixels with a given gray value, as computed by [EasylImage::Histogram](#)).

EasylImage.DilateBox

Performs a dilation on an image (maximum of the gray values in a defined neighborhood) on a rectangular kernel.

[VB6]

```
void DilateBox(
    EROIBW1 sourceImage,
    EROIBW1 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

void DilateBox(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

void DilateBox(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

void DilateBox(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half of the box width minus one (by default, **halfOfKernelWidth = 1; 0** is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as **halfOfKernelWidth; 0** is allowed).

EasylImage.DilateDisk

Performs a dilation on an image (maximum of the gray values in a defined neighborhood) on a quasi-circular kernel.

```
[VB6]

void DilateDisk(
    EROIBW1 sourceImage,
    EROIBW1 destinationImage,
    Long halfOfKernelWidth
)

void DilateDisk(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth
)

void DilateDisk(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Long halfOfKernelWidth
)

void DilateDisk(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Long halfOfKernelWidth
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half width of the kernel minus one (by default, **halfOfKernelWidth =1; 0** is allowed).

EasylImage.Distance

Computes the morphological distance function on a binary image (0 for black, non 0 for white).

[VB6]

```
void Distance(
    EROIBW8 sourceImage,
    EIImageBW16 destinationImage,
    Long valueOutOfImage
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as source image.

valueOutOfImage

Out-of-bounds image value. By default, this value is **0**.

Remarks

So, each pixel of the destination image will contain, at the end of the processing, the morphological distance of the corresponding pixel in the source image. The distance function at a given pixel tells how many erosion passes are required to set it to black.

EasylImage.DoubleThreshold

Converts an image by setting all pixels below the low threshold to a low value, all pixels above the high threshold to a high value, and the remaining pixels to an intermediate value.

[VB6]

```
void DoubleThreshold(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long lowThreshold,
    Long highThreshold,
    Byte lowValue,
    Byte middleValue,
    Byte highValue
)

void DoubleThreshold(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Long lowThreshold,
    Long highThreshold,
    EBW16 lowValue,
    EBW16 middleValue,
    EBW16 highValue
)

void DoubleThreshold(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Long lowThreshold,
    Long highThreshold
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

lowThreshold

Low threshold value.

highThreshold

High threshold value.

lowValue

Value for pixels strictly below the low threshold.

middleValue

Value for pixels that are above or equal to the low threshold, and below the high threshold.

highValue

Value for pixels above or equal to the high threshold.

EasylImage.Equalize

Equalizes an image histogram (the gray levels are re-mapped so that the histogram becomes as close to uniform as possible).

[VB6]

```
void Equalize(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

void Equalize(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

Remarks

This strongly enhances the contrast in dark areas.

EasylImage.ErodeBox

Performs an erosion on an image (minimum of the gray values in a defined neighborhood) on a rectangular kernel.

[VB6]

```
void ErodeBox(
    EROIBW1 sourceImage,
    EROIBW1 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

void ErodeBox(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

void ErodeBox(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

void ErodeBox(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half of the box width minus one (by default, **halfOfKernelWidth = 1**; **0** is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as **halfOfKernelWidth**; **0** is allowed).

EasylImage.ErodeDisk

Performs an erosion on an image (minimum of the gray values in a defined neighborhood) on a quasi-circular kernel.

[VB6]

```
void ErodeDisk(
    EROIBW1 sourceImage,
    EROIBW1 destinationImage,
    Long halfOfKernelWidth
)

void ErodeDisk(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth
)

void ErodeDisk(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Long halfOfKernelWidth
)

void ErodeDisk(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Long halfOfKernelWidth
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half width of the kernel minus one (by default, **halfOfKernelWidth = 1; 0** is allowed).

EasylImage.Focusing

Returns a measure of the focusing of an image by computing the total gradient energy.

```
[VB6]  
Single Focusing(  
    EROIBW8 image  
)  
  
Single Focusing(  
    EROIBW16 image  
)  
  
Single Focusing(  
    EROIC24 image  
)
```

Parameters

image

Pointer to the source image/ROI.

Remarks

When this quantity is maximum for a given image, sharp focusing is achieved.

EasylImage.Gain

Transforms an image, applying a gain and offset to all pixels.

```
[VB6]  
void Gain(  
    EROIC24 sourceImage,  
    EROIC24 destinationImage,  
    EColor Gain  
)
```

Parameters*sourceImage*

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

*Gain*Constant gain. By default (argument omitted) **1**, i.e. no change.**Remarks**

The gain should remain close to **1**, and allows contrast adjustment of the image.

The offset can be positive or negative, and allows to adjust the image intensity. The resulting values are always saturated to range **[0..255]**.

For color images, the separate gain and offset values are specified as triple of values stored in a [EColor](#). The default values leave the image unchanged.

Internally, the computations are achieved through fixed-point arithmetic with 5 bits of precision for the fractional part. This can result in loss of precision with small gains.

Easylimage.GainOffset

Transforms an image, applying a gain and offset to all pixels.

[VB6]

```
void GainOffset(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Single gain,
    Single offset
)

void GainOffset(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Single gain,
    Single offset
)
```

```
void GainOffset(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    EColor Gain,
    EColor Offset
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

gain

Constant gain. By default (argument omitted) **1**, i.e. no change.

offset

Constant offset. By default (argument omitted) **0**, i.e. no change.

Gain

Constant gain. By default (argument omitted) **1**, i.e. no change.

Offset

Constant offset. By default (argument omitted) **0**, i.e. no change.

Remarks

The gain should remain close to **1**, and allows contrast adjustment of the image.

The offset can be positive or negative, and allows to adjust the image intensity. The resulting values are always saturated to range **[0..255]**.

For color images, the separate gain and offset values are specified as triple of values stored in a [EColor](#). The default values leave the image unchanged.

Internally, the computations are achieved through fixed-point arithmetic with 5 bits of precision for the fractional part. This can result in loss of precision with small gains.

Easylimage.GetFrame

Extracts the frame of given parity from an image.

```
[VB6]

void GetFrame(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Boolean odd
)

void GetFrame(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Boolean odd
)

void GetFrame(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Boolean odd
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as the source image.

odd

Specifies which frame is extracted (the frame made up of all lines of the same parity as **odd**).

Remarks

The size of the destination image is determined as follows: **DstImage_Width = SrcImage_Width**
DstImage_Height = (SrcImage_Height + 1 - odd) / 2

EasylImage.GetProfilePeaks

Detects peaks in a gray-level profile. Maxima as well as minima are considered.

```
[VB6]
```

```
void GetProfilePeaks(
    EBW8Vector profile,
    EPeakVector peaks,
    Long lowThreshold,
    Long highThreshold,
    Long minimumAmplitude,
    Long minimumArea
)

void GetProfilePeaks(
    EBW16Vector profile,
    EPeakVector peaks,
    Long lowThreshold,
    Long highThreshold,
    Long minimumAmplitude,
    Long minimumArea
)
```

Parameters

profile

Pointer to the source vector.

peaks

Pointer to the destination vector.

lowThreshold

Threshold used for the minimum peaks.

highThreshold

Threshold used for the maximum peaks.

minimumAmplitude

Minimum amplitude required for a peak to be kept (may be **0**).

minimumArea

Minimum area required for a peak to be kept (may be **0**).

Remarks

To eliminate false peaks due to noise, two selection criteria are used. A peak is the portion of the signal that is above [below] a given threshold. The peak amplitude is defined to be the difference between the threshold value and the maximum [minimum] signal value. The peak area is defined to be the surface comprised between the signal curve and the horizontal line at the given threshold. The result is stored in a peaks vector.

EasylImage.GradientScalar

Computes the (scalar) gradient image derived from a given source image.

```
[VB6]

void GradientScalar(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    EROIBW8 lookupTable
)

void GradientScalar(
    EROIBW32 sourceImage,
    EROIBW8 destinationImage,
    EROIBW8 lookupTable
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

lookupTable

Pointer to the image/ROI used as a preset lookup-table. This lookup table can be generated by one of [EasylImage::ArgumentImage](#) or [EasylImage::ModulusImage](#), or be user-defined.

Remarks

The scalar value derived from the gradient depends on the preset lookup-table image. The gradient of a gray-scale image corresponds to a vector, the components of which are the partial derivatives of the gray-level signal in the horizontal and vertical direction. A vector can be characterized by a direction and a length, corresponding to the gradient orientation, here called *argument*, and the gradient magnitude, here called *magnitude*. Function [GradientScalar](#) generates a gradient direction or gradient magnitude map (gray-level image) from a given gray-level image. For efficiency, a pre-computed lookup-table is used to define the desired transformation. This lookup-table is stored as a standard [EImageBW8/EImageBW16](#). Use one of [EasylImage::ArgumentImage](#) or [EasylImage::ModulusImage](#) once before calling [GradientScalar](#).

EasylImage.GravityCenter

Computes the coordinates of the gravity center of an image, i.e. the average coordinates of the pixels above (or on) the threshold.

[VB6]

```
void GravityCenter(
    EROIBW8 sourceImage,
    Long threshold,
    Single gravityX,
    Single gravityY
)

void GravityCenter(
    EROIBW16 sourceImage,
    Long threshold,
    Single gravityX,
    Single gravityY
)

void GravityCenter(
    EROIBW8 sourceImage,
    EROIBW8 mask,
    Long threshold,
    Single gravityX,
    Single gravityY
)

void GravityCenter(
    EROIBW16 sourceImage,
    EROIBW8 mask,
    Long threshold,
    Single gravityX,
    Single gravityY
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

threshold

Threshold.

gravityX

Reference to the gravity center abscissa.

gravityY

Reference to the gravity center ordinate.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

EasylImage.HDRFusion

Fuses two images using HDR principles.

[VB6]

```
void HDRFusion(
    EROIBW8 darkSrc,
    EROIBW8 lightSrc,
    Long shutterSpeedFactor,
    EROIBW16 dst
)

void HDRFusion(
    EROIBW16 darkSrc,
    EROIBW16 lightSrc,
    Long shutterSpeedFactor,
    EROIBW16 dst
)

void HDRFusion(
    EROIBW16 darkSrc,
    EROIBW16 lightSrc,
    Long shutterSpeedFactor,
    EROIBW32 dst
)

void HDRFusion(
    EROIC24 darkSrc,
    EROIC24 lightSrc,
    Long shutterSpeedFactor,
    EROIC48 dst
)
```

```
void HDRFusion(
    EROIC48 darkSrc,
    EROIC48 lightSrc,
    Long shutterSpeedFactor,
    EROIC48 dst
)
```

Parameters*darkSrc*

Dark input image (high shutter speed).

lightSrc

Light input image (low shutter speed).

shutterSpeedFactor

Shutter speed factor between light and dark image.

dst

Destination image.

EasylImage.Histogram

Computes the histogram of an image (count of each gray-level value).

[VB6]

```
void Histogram(
    EROIBW8 sourceImage,
    EBWHistogramVector histogram
)

void Histogram(
    EROIBW16 sourceImage,
    EBWHistogramVector histogram,
    Long mostSignificantBit,
    Long numberOfSignificantBits,
    Boolean saturate
)
```

```

void Histogram(
    EROIBW32 sourceImage,
    EBWHistogramVector histogram,
    Long mostSignificantBit,
    Long numberOfSignificantBits,
    Boolean saturate
)

void Histogram(
    EROIBW8 sourceImage,
    EROIBW8 mask,
    EBWHistogramVector histogram
)

void Histogram(
    EROIBW16 sourceImage,
    EROIBW8 mask,
    EBWHistogramVector histogram,
    Long mostSignificantBit,
    Long numberOfSignificantBits,
    Boolean saturate
)

void Histogram(
    EROIBW32 sourceImage,
    EROIBW8 mask,
    EBWHistogramVector histogram,
    Long mostSignificantBit,
    Long numberOfSignificantBits,
    Boolean saturate
)

```

Parameters

sourceImage

Pointer to the source image/ROI.

histogram

Pointer to the destination vector.

mostSignificantBit

Index of the most significant bit of the pixels (**0** has weight 1).

numberOfSignificantBits

Number of significant bits; the histogram will possess $2^{\text{numberOfSignificantBits}}$ entries.

saturate

Boolean indicating if values larger than $2^{\text{mostSignificantBit}} - 1$ are saturated (default **TRUE**) or not (**FALSE**).

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

EasylImage.HistogramThreshold

Determines an appropriate threshold level based on the histogram contents, using an automatic threshold mode.

[VB6]

```
void HistogramThreshold(
    EBWHistogramVector histogram,
    Long threshold,
    Single averageOfPixelsBelowThreshold,
    Single averageOfPixelsAboveThreshold,
    Single relativeThreshold,
    Long from,
    Long to
)
```

Parameters

histogram

Pointer to a vector containing an image histogram.

threshold

reference to the threshold value. Before calling the function, must be set to the appropriate thresholding mode, as defined by [EThresholdMode](#).

averageOfPixelsBelowThreshold

Average gray level of the dark pixels (below threshold).

averageOfPixelsAboveThreshold

Average gray level of the light pixels (above threshold).

relativeThreshold

Relative threshold value, relevant only in the [EThresholdMode_Relative](#) mode.

from

Lower bound of the analyzed gray-level range.

to

Upper bound of the analyzed gray-level range.

Remarks

Additionally, returns the average gray levels in the regions below and above the threshold. The threshold level can be computed by analyzing a range of gray levels in the histogram.

EasylImage.HistogramThresholdBW16

Determines an appropriate threshold level based on the histogram contents, using an automatic threshold mode.

[VB6]

```
void HistogramThresholdBW16(
    EBWHistogramVector histogram,
    Long threshold,
    Single averageOfPixelsBelowThreshold,
    Single averageOfPixelsAboveThreshold,
    Single relativeThreshold,
    Long from,
    Long to
)
```

Parameters

histogram

Pointer to a vector containing an image histogram.

threshold

reference to the threshold value. Before calling the function, must be set to the appropriate thresholding mode, as defined by [EThresholdMode](#).

averageOfPixelsBelowThreshold

Average gray level of the dark pixels (below threshold).

averageOfPixelsAboveThreshold

Average gray level of the light pixels (above threshold).

relativeThreshold

Relative threshold value, relevant only in the [EThresholdMode_Relative](#) mode.

from

Lower bound of the analyzed gray-level range.

to

Upper bound of the analyzed gray-level range.

Remarks

Additionally, returns the average gray levels in the regions below and above the threshold. The threshold level can be computed by analyzing a range of gray levels in the histogram.

EasylImage.HitAndMiss

Apply the morphological hit-and-miss transform to detect a particular pattern of foreground and background pixels in a BW8, BW16 or C24 image/ROI.

```
[VB6]

void HitAndMiss(
    EROIBW8 source,
    EROIBW8 destination,
    EHItAndMissKernel kernel
)

void HitAndMiss(
    EROIBW16 source,
    EROIBW16 destination,
    EHItAndMissKernel kernel
)

void HitAndMiss(
    EROIC24 source,
    EROIC24 destination,
    EHItAndMissKernel kernel
)
```

Parameters

source

The source image/ROI.

destination

The destination image/ROI.

kernel

The hit-and-miss kernel.

EasylImage.HorizontalMirror

Mirrors an image horizontally (the columns are swapped).

```
[VB6]  
  
void HorizontalMirror(  
    EROIBW8 sourceImage  
)  
  
void HorizontalMirror(  
    EROIC24 sourceImage  
)  
  
void HorizontalMirror(  
    EROIBW16 sourceImage  
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

EasylImage.ImageToLineSegment

Copies the pixel values along a given line segment (arbitrarily oriented) to a vector.

```
[VB6]  
  
void ImageToLineSegment(  
    EROIBW8 sourceImage,  
    EBW8Vector path,  
    Long X0,  
    Long Y0,  
    Long X1,  
    Long Y1  
)
```

```
void ImageToLineSegment(
    EROIBW16 sourceImage,
    EBW16Vector path,
    Long X0,
    Long Y0,
    Long X1,
    Long Y1
)

void ImageToLineSegment(
    EROIC24 sourceImage,
    EC24Vector path,
    Long X0,
    Long Y0,
    Long X1,
    Long Y1
)

void ImageToLineSegment(
    EROIBW8 sourceImage,
    EROIBW8 mask,
    EBW8 outOfMaskValue,
    EBW8Vector path,
    Long X0,
    Long Y0,
    Long X1,
    Long Y1
)

void ImageToLineSegment(
    EROIBW16 sourceImage,
    EROIBW8 mask,
    EBW16 outOfMaskValue,
    EBW16Vector path,
    Long X0,
    Long Y0,
    Long X1,
    Long Y1
)
```

```
void ImageToLineSegment(
    EROIC24 sourceImage,
    EROIBW8 mask,
    EC24 outOfMaskValue,
    EC24Vector path,
    Long X0,
    Long Y0,
    Long X1,
    Long Y1
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

path

Pointer to the destination vector.

X0

Coordinates of the starting point of the segment.

Y0

Coordinates of the starting point of the segment.

X1

Coordinates of the ending point of the segment.

Y1

Coordinates of the ending point of the segment.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

outOfMaskValue

The value to be given to the pixels that lie out of the mask.

Remarks

The line segment must be wholly contained within the image. The vector length is adjusted automatically.

Easylimage.ImageToPath

Given a path described by coordinates in a path vector, copies the corresponding pixel values into the same vector.

[VB6]

```
void ImageToPath(
    EROIBW8 sourceImage,
    EBW8PathVector path
)

void ImageToPath(
    EROIBW16 sourceImage,
    EBW16PathVector path
)

void ImageToPath(
    EROIC24 sourceImage,
    EC24PathVector path
)

void ImageToPath(
    EROIBW8 sourceImage,
    EROIBW8 mask,
    EBW8 outOfMaskValue,
    EBW8PathVector path
)

void ImageToPath(
    EROIBW16 sourceImage,
    EROIBW8 mask,
    EBW16 outOfMaskValue,
    EBW16PathVector path
)

void ImageToPath(
    EROIC24 sourceImage,
    EROIBW8 mask,
    EC24 outOfMaskValue,
    EC24PathVector path
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

path

Pointer to the destination vector.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

outOfMaskValue

The value to be given to the pixels that lie out of the mask.

EasylImage.IsodataThreshold

Computes a suitable threshold value for a histogram.

[VB6]

```
EBW8 IsodataThreshold(
    EBWHistogramVector histogram,
    Long from,
    Long to
)
```

Parameters

histogram

Pointer to a vector containing an image histogram.

from

Lower bound of the useful gray-level interval (by default, **0**).

to

Upper bound of the useful gray-level interval (by default, **255**).

Remarks

The "isodata" rule is used: if one computes the average gray level of pixels below the threshold and the average gray level of pixels above the threshold, the threshold lies exactly halfway between them. Optionally, the threshold selection can be restricted to a range of gray-level values.

It is possible that, in the automatic or relative thresholding modes, the computed threshold exceeds the dynamic range of the return type. In this case, the value is clipped to the maximum value that is representable in the return type.

EasylImage.IsodataThresholdBW16

Computes a suitable threshold value for a histogram.

[VB6]

```
EBW16 IsodataThresholdBW16(
    EBWHistogramVector histogram,
    Long from,
    Long to
)
```

Parameters

histogram

Pointer to a vector containing an image histogram.

from

Lower bound of the useful gray-level interval (by default, **0**).

to

Upper bound of the useful gray-level interval (by default, **65535**).

Remarks

The "isodata" rule is used: if one computes the average gray level of pixels below the threshold and the average gray level of pixels above the threshold, the threshold lies exactly halfway between them. Optionally, the threshold selection can be restricted to a range of gray-level values. Returns the threshold.

EasylImage.LinearTransform

Applies a general affine transformation.

[VB6]

```
void LinearTransform(
    EROIBW8 sourceImage,
    Single Axx,
    Single Axy,
    Single Ax,
    Single Ayx,
    Single Ayy,
    Single Ay,
    EROIBW8 destinationImage,
    Long interpolationBits
)
```

```
void LinearTransform(
    EROIBW16 sourceImage,
    Single Axx,
    Single Axy,
    Single Ax,
    Single Ayx,
    Single Ayy,
    Single Ay,
    EROIBW16 destinationImage,
    Long interpolationBits
)

void LinearTransform(
    EROIC24 sourceImage,
    Single Axx,
    Single Axy,
    Single Ax,
    Single Ayx,
    Single Ayy,
    Single Ay,
    EROIC24 destinationImage,
    Long interpolationBits
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

Axx

See formula below.

Axy

See formula below.

Ax

See formula below.

Ayx

See formula below.

Ayy

See formula below.

Ay

See formula below.

destinationImage

Pointer to the destination image/ROI.

interpolationBits

Number of bits of accuracy for interpolation. Allowed values are **0** (no interpolation, nearest neighbor), **4** or **8**.

Remarks

An affine transformation is an important class of linear 2D geometric transformations which maps variables (e.g. pixel intensity values located at position (X_{Src} , Y_{Src}) in an input image) into new variables (e.g. (X_{Dst} , Y_{Dst}) in an output image) by applying a linear combination of translation, rotation, scaling and/or shearing (i.e. non-uniform scaling in some directions) operations. The parameters of the [LinearTransform](#) function are the coefficients of the affine equations below:

$$X_{Dst} = A_{xx}X_{Src} + A_{xy}Y_{Src} + A_x$$

$$Y_{Dst} = A_{yx}X_{Src} + A_{yy}Y_{Src} + A_y$$

EasylImage.LineSegmentToImage

Copies the pixel values from a vector or a constant to the pixels of a given line segment (arbitrarily oriented).

[VB6]

```
void LineSegmentToImage(
    EROIBW8 destinationImage,
    EBW8 pixel,
    Long X0,
    Long Y0,
    Long X1,
    Long Y1
)

void LineSegmentToImage(
    EROIBW16 destinationImage,
    EBW16 pixel,
    Long X0,
    Long Y0,
    Long X1,
    Long Y1
)
```

```
void LineSegmentToImage(
    EROIC24 destinationImage,
    EC24 pixel,
    Long X0,
    Long Y0,
    Long X1,
    Long Y1
)

void LineSegmentToImage(
    EROIBW8 destinationImage,
    EBW8Vector path,
    Long X0,
    Long Y0,
    Long X1,
    Long Y1
)

void LineSegmentToImage(
    EROIBW16 destinationImage,
    EBW16Vector path,
    Long X0,
    Long Y0,
    Long X1,
    Long Y1
)

void LineSegmentToImage(
    EROIC24 destinationImage,
    EC24Vector path,
    Long X0,
    Long Y0,
    Long X1,
    Long Y1
)
```

Parameters

destinationImage

Pointer to the destination image/ROI.

pixel

Constant color value.

X0

Coordinates of the starting point of the segment.

Y0

Coordinates of the starting point of the segment.

X1

Coordinates of the ending point of the segment.

Y1

Coordinates of the ending point of the segment.

path

Pointer to the source vector.

Remarks

The line segment must be wholly contained within the image.

EasylImage.LocalAverage

Computes the average in a rectangular window centered on every pixel.

```
[VB6]

void LocalAverage(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfWidth,
    Long halfHeight
)

void LocalAverage(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Long halfWidth,
    Long halfHeight
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

halfWidth

Half of the window width minus one.

halfHeight

Half of the window height minus one.

Remarks

The window dimensions can be an arbitrary odd integer. The running time of this function does not depend on the window size.

EasylImage.LocalDeviation

Computes the standard deviation in a rectangular window centered on every pixel.

```
[VB6]

void LocalDeviation(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfWidth,
    Long halfHeight
)

void LocalDeviation(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Long halfWidth,
    Long halfHeight
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as the source image.

halfWidth

Half of the window width minus one.

halfHeight

Half of the window height minus one.

Remarks

The window dimensions can be an arbitrary odd integer. The running time of this function does not depend on the window size.

EasylImage.Lut

Transforms the gray levels of an image, using a lookup table stored in a vector (of unsigned values).

[VB6]

```
void Lut(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    EBW16Vector lookupTable
)

void Lut(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    EBW8Vector lookupTable
)

void Lut(
    EROIBW16 sourceImage,
    EROIBW8 destinationImage,
    EBW8Vector lookupTable,
    Long numberOfScalingBits
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

lookupTable

Pointer to the lookup vector.

numberOfScalingBits

Number of scaling bits (or right padding bits).

Remarks

A 16-bit image usually does not make use of its 16 bits. In most cases, only 10 or 12 bits are used. These bits are called *significant bits*. In the 16-bit information, significant bits can be left aligned, right aligned or not aligned at all. To indicate which are the significant bits, we have to tell how many bits are significant and the number of right padding bits (0 right padding bit means that significant bits are right aligned). The number of significant bits is given by the number of Look Up table entries. For example a Lut of 1024 entries is used for an image of 10 significant bits (as $2^{10} = 1024$). The number of right padding bits is given by means of the **numberOfScalingBits** parameter. Leaving this parameter undefined indicates that the significant bits are left aligned on the word.

EasylImage.MatchFrames

Determines the optimal shift amplitude by comparing two successive lines of the image.

```
[VB6]

void MatchFrames (
    EROIBW8 sourceImage,
    Long fixedRow,
    Long minimumOffset,
    Long maximumOffset,
    Long bestOffset
)

void MatchFrames (
    EROIBW16 sourceImage,
    Long fixedRow,
    Long minimumOffset,
    Long maximumOffset,
    Long bestOffset
)

void MatchFrames (
    EROIC24 sourceImage,
    Long fixedRow,
    Long minimumOffset,
    Long maximumOffset,
    Long bestOffset
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

fixedRow

Index of the line used for comparison. Line **fixedRow** remains in place and is compared with line (**fixedRow + 1**), shifted by some amount.

minimumOffset

Minimum value of the allowed offset (positive to the right).

maximumOffset

Maximum value of the allowed offset (positive to the right).

bestOffset

Estimated shift amplitude.

Remarks

These lines should be chosen such that they cross some edges or non-uniform areas. When an image is interlaced, the two frames (even and odd lines) are not recorded at the same time. If there is movement in the scene, a visible artifact can result (the edges of objects exhibit a "comb" effect). When the movement is uniform and horizontal (objects on a conveyor belt), one cure to this problem is to shift one of the frames horizontally with respect to the other frame (using [EasylImage::RealignFrame](#)). The amplitude of the shift can be estimated automatically.

EasylImage.Median

Applies a median filter to an image (median of the gray values in a 3x3 neighborhood).

[VB6]

```
void Median(
    EROIBW1 sourceImage,
    EROIBW1 destinationImage
)

void Median(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

void Median(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)
```

```
void Median(
    EROIC24 sourceImage,
    EROIC24 destinationImage
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as source image.

EasylImage.ModulusImage

Prepares a lookup-table image for use for gradient magnitude computation.

[VB6]

```
void ModulusImage(
    EIImageBW8 destinationImage,
    Single gain
)
```

Parameters

destinationImage

Pointer to the destination image.

gain

Gain value to be applied to the modulus. **1** saturates; **1/Sqrt(2)** does not.

Remarks

The modulus of the gradient argument can be adjusted to avoid saturation. [ModulusImage](#) sets a lookup-table image for use with function [EasylImage::GradientScalar](#), ready to compute the modulus of the gradient in the source image, i.e. its amplitude (as defined by the Euclidian norm). The argument will be returned as a value in range **0..255** suitable for storage in an [EIImageBW8](#) or as a value in range **0..65535** suitable for storage in an [EIImageBW16](#). A gain coefficient can be adjusted to avoid saturation (gain = 1 saturates gradient amplitudes larger than 255 in the [EBW8](#) case and 65535 in the [EBW16](#) case; gain = **1/Sqrt(2)** never saturates).

EasylImage.MorphoGradientBox

Computes the morphological gradient of an image using a rectangular kernel.

[VB6]

```
void MorphoGradientBox(
    EROIBW1 sourceImage,
    EROIBW1 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

void MorphoGradientBox(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

void MorphoGradientBox(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

void MorphoGradientBox(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half of the box width minus one (by default, **halfOfKernelWidth = 1**; **0** is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as **halfOfKernelWidth**; **0** is allowed).

Remarks

The morphological gradient is the difference between the dilation and the erosion of the image, using the same structuring element. The kernel size is a pair of odd numbers; they must be halved before they are passed. For instance, a 3x5 kernel is passed as 1x2.

EasylImage.MorphoGradientDisk

Computes the morphological gradient of an image using a quasi-circular kernel.

[VB6]

```
void MorphoGradientDisk(
    EROIBW1 sourceImage,
    EROIBW1 destinationImage,
    Long halfOfKernelWidth
)

void MorphoGradientDisk(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth
)

void MorphoGradientDisk(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Long halfOfKernelWidth
)

void MorphoGradientDisk(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Long halfOfKernelWidth
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half width of the kernel minus one (by default, **halfOfKernelWidth = 1**; **0** is allowed).

Remarks

The morphological gradient is the difference between the dilation and the erosion of the image, using the same structuring element.

EasylImage.Normalize

Normalizes an image, i.e. applies a linear transform to the gray levels so that their average and standard deviation are imposed.

[VB6]

```
void Normalize(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Single imposedAverage,
    Single imposedStandardDeviation
)

void Normalize(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Single imposedAverage,
    Single imposedStandardDeviation
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

imposedAverage

Imposed average.

imposedStandardDeviation

Imposed standard deviation.

EasylImage.Offset

Transforms an image, applying a gain and offset to all pixels.

[VB6]

```
void Offset(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    EColor Offset
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

Offset

Constant offset. By default (argument omitted) **0**, i.e. no change.

Remarks

The gain should remain close to **1**, and allows contrast adjustment of the image.

The offset can be positive or negative, and allows to adjust the image intensity. The resulting values are always saturated to range **[0..255]**.

For color images, the separate gain and offset values are specified as triple of values stored in a **EColor**. The default values leave the image unchanged.

Internally, the computations are achieved through fixed-point arithmetic with 5 bits of precision for the fractional part. This can result in loss of precision with small gains.

EasylImage.OpenBox

Performs an opening on an image (erosion followed by dilation) on a rectangular kernel.

[VB6]

```

void OpenBox(
    EROIBW1 sourceImage,
    EROIBW1 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

void OpenBox(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

void OpenBox(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

void OpenBox(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

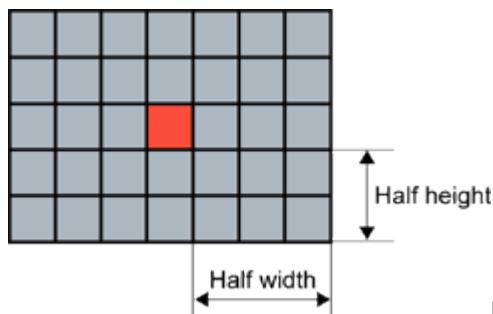
Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half of the box width minus one, as shown on the picture below (by default, **halfOfKernelWidth =1**; **0** is allowed).

halfOfKernelHeight

Half of the box height minus one, as shown on the picture below (by default, same as **halfOfKernelWidth**; **0** is allowed).

Remarks

Rectangular kernel of half width = 3 and half height = 2

Easylimage.OpenDisk

Performs an opening on an image (erosion followed by dilation) on a quasi-circular kernel.

[VB6]

```

void OpenDisk(
    EROIBW1 sourceImage,
    EROIBW1 destinationImage,
    Long halfOfKernelWidth
)

void OpenDisk(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth
)

void OpenDisk(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Long halfOfKernelWidth
)

void OpenDisk(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Long halfOfKernelWidth
)

```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If **NULL** (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half width of the kernel minus one, as shown on the picture below (by default, **halfOfKernelWidth = 1; 0** is allowed).

Easylimage.Oper

Applies the desired arithmetic or logic pixel-wise operator between two images or constants.

[VB6]

```
void Oper(
    EAriithmeticLogicOperation operation,
    EBW1 constant,
    EROIBW1 destinationImage
)

void Oper(
    EAriithmeticLogicOperation operation,
    EROIBW1 sourceImage,
    EROIBW1 destinationImage
)

void Oper(
    EAriithmeticLogicOperation operation,
    EROIBW1 sourceImage0,
    EROIBW1 sourceImage1,
    EROIBW1 destinationImage
)

void Oper(
    EAriithmeticLogicOperation operation,
    EBW8 constant,
    EROIBW8 destinationImage
)

void Oper(
    EAriithmeticLogicOperation operation,
    EBW16 constant,
    EROIBW16 destinationImage
)
```

```
void Oper(
    EAithmeticLogicOperation operation,
    EC24 constant,
    EROIC24 destinationImage
)

void Oper(
    EAithmeticLogicOperation operation,
    EBW8 constant,
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

void Oper(
    EAithmeticLogicOperation operation,
    EBW16 constant,
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)

void Oper(
    EAithmeticLogicOperation operation,
    EC24 constant,
    EROIC24 sourceImage,
    EROIC24 destinationImage
)

void Oper(
    EAithmeticLogicOperation operation,
    EROIBW8 sourceImage,
    EBW8 constant,
    EROIBW8 destinationImage
)

void Oper(
    EAithmeticLogicOperation operation,
    EROIBW16 sourceImage,
    EBW16 constant,
    EROIBW16 destinationImage
)

void Oper(
    EAithmeticLogicOperation operation,
    EROIC24 sourceImage,
    EC24 constant,
    EROIC24 destinationImage
)
```

```
void Oper(
    EAithmeticLogicOperation operation,
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

void Oper(
    EAithmeticLogicOperation operation,
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)

void Oper(
    EAithmeticLogicOperation operation,
    EROIC24 sourceImage,
    EROIC24 destinationImage
)

void Oper(
    EAithmeticLogicOperation operation,
    EROIBW8 sourceImage,
    EROIC24 destinationImage
)

void Oper(
    EAithmeticLogicOperation operation,
    EROIBW8 sourceImage0,
    EROIBW8 sourceImage1,
    EROIBW8 destinationImage
)

void Oper(
    EAithmeticLogicOperation operation,
    EROIBW16 sourceImage0,
    EROIBW16 sourceImage1,
    EROIBW16 destinationImage
)

void Oper(
    EAithmeticLogicOperation operation,
    EROIC24 sourceImage0,
    EROIC24 sourceImage1,
    EROIC24 destinationImage
)
```

```
void Oper(
    EAithmeticLogicOperation operation,
    EROIBW8 sourceImage0,
    EROIBW8 sourceImage1,
    EROIBW16 destinationImage
)

void Oper(
    EAithmeticLogicOperation operation,
    EROIBW16 sourceImage0,
    EROIBW8 sourceImage1,
    EROIBW16 destinationImage
)

void Oper(
    EAithmeticLogicOperation operation,
    EROIBW8 sourceImage0,
    EROIBW8 sourceImage1,
    EROIC24 destinationImage
)

void Oper(
    EAithmeticLogicOperation operation,
    EROIBW8 sourceImage0,
    EROIC24 sourceImage1,
    EROIC24 destinationImage
)

void Oper(
    EAithmeticLogicOperation operation,
    EROIC24 sourceImage0,
    EROIBW8 sourceImage1,
    EROIC24 destinationImage
)
```

Parameters

operation

Arithmetic or logic operator, as defined by [EAithmeticLogicOperation](#).

constant

Gray-level or color constant.

destinationImage

Pointer to the destination image/ROI.

sourceImage

Pointer to the second source image/ROI (right operand).

sourceImage0

Pointer to the first source image/ROI (left operand).

sourceImage1

Pointer to the second source image/ROI (right operand).

Remarks

The source and destination images may be the same. When the source operands are two color images/constants, the components are combined pair-wise; the result is a color image. When the source operands are a color image and a gray-level image, each color component is combined with the gray-level component. The result is a color image.

EasylImage.Overlay

Overlays an image on the top of a color image, at a given position.

[VB6]

```
void Overlay(
    EROIC24 sourceImage,
    EROIC16 destinationImage,
    Long panX,
    Long panY,
    EC24 referenceValue
)

void Overlay(
    EROIC24 sourceImage,
    EROIC15 destinationImage,
    Long panX,
    Long panY,
    EC24 referenceValue
)

void Overlay(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Long panX,
    Long panY,
    EC24 referenceValue
)
```

```
void Overlay(
    EROIC24 sourceImage,
    EROIBW8 mask,
    EROIC15 destinationImage,
    Long panX,
    Long panY
)

void Overlay(
    EROIC24 sourceImage,
    EROIBW8 mask,
    EROIC16 destinationImage,
    Long panX,
    Long panY
)

void Overlay(
    EROIC24 sourceImage,
    EROIBW8 mask,
    EROIC24 destinationImage,
    Long panX,
    Long panY
)

void Overlay(
    EROIBW8 sourceImage,
    EROIC24 overlay,
    EROIC24 destinationImage,
    Long panX,
    Long panY,
    EC24 referenceValue
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

panX

Translation for panning in the horizontal direction, in pixels.

panY

Translation for panning in the vertical direction, in pixels.

referenceValue

Reference color.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

overlay

When a **BW8** source image is specified, pointer to the overlay image/ROI.

Remarks

If a color image is provided as the source image, all the pixels of this image are copied to the destination image, but the ones that equal the reference color. If a **BW8** image is provided as the source image, all the pixels of the overlay image are copied to the destination image, but the ones that equal the reference color, the latter being replaced by the content of the source image.

EasylImage.OverlayColor

Sets the color of the overlay in the destination image when a **BW8** Image is used as overlay source image in functions:

[VB6]

OverlayColor As EC24

read-write

Remarks

Note. When a **C24** Image is used as overlay source image, the color of the overlay in destination image is the same as the one in the overlay source image, thus allowing multi colored overlays.

EasylImage.PathToImage

Given a path described by coordinates in a path vector, copies the pixel values from the path vector to the corresponding image pixels.

[VB6]

```
void PathToImage(
    EROIBW8 sourceImage,
    EBW8PathVector path
)

void PathToImage(
    EROIBW16 sourceImage,
    EBW16PathVector path
)

void PathToImage(
    EROIC24 sourceImage,
    EC24PathVector path
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

path

Pointer to the destination vector.

EasylImage.PixelAverage

Computes the average pixel value in a gray-level or color image.

[VB6]

```
void PixelAverage(
    EROIBW8 sourceImage,
    Single average
)

void PixelAverage(
    EROIBW16 sourceImage,
    Single average
)
```

```
void PixelAverage(
    EROIC24 sourceImage,
    Single average0,
    Single average1,
    Single average2
)

void PixelAverage(
    EROIBW8 sourceImage,
    EROIBW8 inputMask,
    Single average
)

void PixelAverage(
    EROIBW16 sourceImage,
    EROIBW8 inputMask,
    Single average
)

void PixelAverage(
    EROIC24 sourceImage,
    EROIBW8 inputMask,
    Single average0,
    Single average1,
    Single average2
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

average

Reference to the average gray-level value.

average0

Reference to the average values for the first color channel.

average1

Reference to the average values for the second color channel.

average2

Reference to the average values for the third color channel.

inputMask

Pointer to the mask, which allows functions to be applied on a particular region in the image.

EasylImage.PixelCompare

Counts the number of pixels differing between two images.

[VB6]

```
Long PixelCompare(
    EROIBW1 sourceImage0,
    EROIBW1 sourceImage1
)

Long PixelCompare(
    EROIBW8 sourceImage0,
    EROIBW8 sourceImage1
)

Long PixelCompare(
    EROIBW16 sourceImage0,
    EROIBW16 sourceImage1
)

Long PixelCompare(
    EROIC24 sourceImage0,
    EROIC24 sourceImage1
)

Long PixelCompare(
    EROIBW8 sourceImage0,
    EROIBW8 sourceImage1,
    EROIBW8 mask
)

Long PixelCompare(
    EROIBW16 sourceImage0,
    EROIBW16 sourceImage1,
    EROIBW8 mask
)

Long PixelCompare(
    EROIC24 sourceImage0,
    EROIC24 sourceImage1,
    EROIBW8 mask
)
```

Parameters

sourceImage0

Pointer to the first source image/ROI.

sourceImage1

Pointer to the second source image/ROI.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

EasylImage.PixelCount

Counts the pixels in the three value classes separated by two thresholds.

[VB6]

```
void PixelCount(
    EROIBW8 sourceImage,
    EBW8 lowThreshold,
    EBW8 highThreshold,
    Long numberOfPixelsBelowThreshold,
    Long numberOfPixelsBetweenThresholds,
    Long numberOfPixelsAboveThreshold
)

void PixelCount(
    EROIBW16 sourceImage,
    EBW16 lowThreshold,
    EBW16 highThreshold,
    Long numberOfPixelsBelowThreshold,
    Long numberOfPixelsBetweenThresholds,
    Long numberOfPixelsAboveThreshold
)

void PixelCount(
    EROIBW8 sourceImage,
    EBW8 lowThreshold,
    EBW8 highThreshold,
    Unsupported variant type numberOfPixelsBelowThreshold,
    Unsupported variant type numberOfPixelsBetweenThresholds,
    Unsupported variant type numberOfPixelsAboveThreshold
)
```

```
void PixelCount(
    EROIBW16 sourceImage,
    EBW16 lowThreshold,
    EBW16 highThreshold,
    Unsupported variant type numberOfPixelsBelowThreshold,
    Unsupported variant type numberOfPixelsBetweenThresholds,
    Unsupported variant type numberOfPixelsAboveThreshold
)

void PixelCount(
    EROIBW8 sourceImage,
    EROIBW8 mask,
    EBW8 lowThreshold,
    EBW8 highThreshold,
    Long numberOfPixelsBelowThreshold,
    Long numberOfPixelsBetweenThresholds,
    Long numberOfPixelsAboveThreshold
)

void PixelCount(
    EROIBW16 sourceImage,
    EROIBW8 mask,
    EBW16 lowThreshold,
    EBW16 highThreshold,
    Long numberOfPixelsBelowThreshold,
    Long numberOfPixelsBetweenThresholds,
    Long numberOfPixelsAboveThreshold
)

void PixelCount(
    EROIBW8 sourceImage,
    EROIBW8 mask,
    EBW8 lowThreshold,
    EBW8 highThreshold,
    Unsupported variant type numberOfPixelsBelowThreshold,
    Unsupported variant type numberOfPixelsBetweenThresholds,
    Unsupported variant type numberOfPixelsAboveThreshold
)

void PixelCount(
    EROIBW16 sourceImage,
    EROIBW8 mask,
    EBW16 lowThreshold,
    EBW16 highThreshold,
    Unsupported variant type numberOfPixelsBelowThreshold,
    Unsupported variant type numberOfPixelsBetweenThresholds,
    Unsupported variant type numberOfPixelsAboveThreshold
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

lowThreshold

Inferior threshold.

highThreshold

Superior threshold.

numberOfPixelsBelowThreshold

Reference to the count of pixels strictly below the inferior threshold.

numberOfPixelsBetweenThresholds

Reference to the count of pixels above or equal to the inferior threshold, and strictly below the superior threshold.

numberOfPixelsAboveThreshold

Reference to the count of pixels above or equal to the superior threshold.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

Easylimage.PixelMax

Computes the maximum gray-level value in an image.

[VB6]

```
void PixelMax(
    EROIBW8 sourceImage,
    EBW8 maximumValue
)

void PixelMax(
    EROIBW16 sourceImage,
    EBW16 maximumValue
)

void PixelMax(
    EROIBW8 sourceImage,
    EROIBW8 mask,
    EBW8 maximumValue
)
```

```
void PixelMax(
    EROIBW16 sourceImage,
    EROIBW8 mask,
    EBW16 maximumValue
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

maximumValue

Reference to the maximum value.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

EasylImage.PixelMaxBW16

Computes the maximum gray-level value in an image.

[VB6]

```
void PixelMaxBW16(
    EROIBW16 sourceImage,
    EBW16 maximumValue
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

maximumValue

Reference to the maximum value.

EasylImage.PixelMaxBW8

Computes the maximum gray-level value in an image.

[VB6]

```
void PixelMaxBW8(
    EROIBW8 sourceImage,
    EBW8 maximumValue
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

maximumValue

Reference to the maximum value.

EasylImage.PixelMin

Computes the minimum gray-level value in an image.

[VB6]

```
void PixelMin(
    EROIBW8 sourceImage,
    EBW8 minValue
)

void PixelMin(
    EROIBW16 sourceImage,
    EBW16 minValue
)
```

```
void PixelMin(
    EROIBW8 sourceImage,
    EROIBW8 mask,
    EBW8 minValue
)

void PixelMin(
    EROIBW16 sourceImage,
    EROIBW8 mask,
    EBW16 minValue
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

minValue

Reference to the minimum value.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

Easylimage.PixelMinBW16

Computes the minimum gray-level value in an image.

[VB6]

```
void PixelMinBW16(
    EROIBW16 sourceImage,
    EBW16 minValue
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

minValue

Reference to the minimum value.

EasylImage.PixelMinBW8

Computes the minimum gray-level value in an image.

```
[VB6]  
void PixelMinBW8(  
    EROIBW8 sourceImage,  
    EBW8 minValue  
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

minValue

Reference to the minimum value.

EasylImage.PixelStat

Computes the minimum, maximum and average gray-level values in an image.

```
[VB6]  
void PixelStat(  
    EROIBW8 sourceImage,  
    EBW8 minValue,  
    EBW8 maxValue,  
    Single average  
)  
  
void PixelStat(  
    EROIBW16 sourceImage,  
    EBW16 minValue,  
    EBW16 maxValue,  
    Single average  
)
```

```
void PixelStat(
    EROIBW8 sourceImage,
    EROIBW8 mask,
    EBW8 minValue,
    EBW8 maxValue,
    Single average
)

void PixelStat(
    EROIBW16 sourceImage,
    EROIBW8 mask,
    EBW16 minValue,
    EBW16 maxValue,
    Single average
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

minValue

Reference to the minimum value.

maxValue

Reference to the maximum value.

average

Reference to the average value.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

Easylimage.PixelStatBW16

Computes the minimum, maximum and average gray-level values in an image.

[VB6]

```
void PixelStatBW16(
    EROIBW16 sourceImage,
    EBW16 minValue,
    EBW16 maxValue,
    Single average
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

minValue

Reference to the minimum value.

maxValue

Reference to the maximum value.

average

Reference to the average value.

EasylImage.PixelStatBW8

Computes the minimum, maximum and average gray-level values in an image.

[VB6]

```
void PixelStatBW8(
    EROIBW8 sourceImage,
    EBW8 minValue,
    EBW8 maxValue,
    Single average
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

minValue

Reference to the minimum value.

maxValue

Reference to the maximum value.

average

Reference to the average value.

EasylImage.PixelStdDev

Computes the average gray-level or color value in an image, the standard deviation of the color components, and the correlation between the color components (in the case of color images).

[VB6]

```
void PixelStdDev(
    EROIBW8 sourceImage,
    Single standardDeviation,
    Single mean
)

void PixelStdDev(
    EROIBW16 sourceImage,
    Single standardDeviation,
    Single mean
)

void PixelStdDev(
    EROIC24 sourceImage,
    Single standardDeviation0,
    Single standardDeviation1,
    Single standardDeviation2,
    Single correlation01,
    Single correlation12,
    Single correlation20,
    Single mean0,
    Single mean1,
    Single mean2
)

void PixelStdDev(
    EROIBW8 sourceImage,
    EROIBW8 mask,
    Single standardDeviation,
    Single mean
)
```

```

void PixelStdDev(
    EROIBW16 sourceImage,
    EROIBW8 mask,
    Single standardDeviation,
    Single mean
)

void PixelStdDev(
    EROIC24 sourceImage,
    EROIBW8 mask,
    Single standardDeviation0,
    Single standardDeviation1,
    Single standardDeviation2,
    Single correlation01,
    Single correlation12,
    Single correlation20,
    Single mean0,
    Single mean1,
    Single mean2
)

```

Parameters

sourceImage

Pointer to the source image/ROI.

standardDeviation

Reference to a variable in which the standard deviation of the pixel values is to be stored (for gray-level images).

mean

Reference to a variable in which the average value of the pixels is to be stored (for gray-level images).

standardDeviation0

Reference to a variable in which the standard deviation of the values of the first color component is to be stored (for color images).

standardDeviation1

Reference to a variable in which the standard deviation of the values of the second color component is to be stored (for color images).

standardDeviation2

Reference to a variable in which the standard deviation of the values of the third color component is to be stored (for color images).

correlation01

Reference to a variable in which the correlation between the values of the first color component and the second color component is to be stored (for color images).

correlation12

Reference to a variable in which the correlation between the values of the second color component and the third color component is to be stored (for color images).

correlation20

-

mean0

Reference to a variable in which the average value of the first color component is to be stored (for color images).

mean1

Reference to a variable in which the average value of the second color component is to be stored (for color images).

mean2

Reference to a variable in which the average value of the third color component is to be stored (for color images).

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

Remarks

The variance can be obtained from the standard deviation by squaring it.

EasylImage.PixelVariance

For a gray-level image, computes the mean and variance of the pixel values.

[VB6]

```
void PixelVariance(
    EROIBW8 sourceImage,
    Single variance,
    Single mean
)

void PixelVariance(
    EROIBW16 sourceImage,
    Single variance,
    Single mean
)
```

```
void PixelVariance(
    EROIC24 sourceImage,
    Single variance0,
    Single variance1,
    Single variance2,
    Single covariance01,
    Single covariance12,
    Single covariance20,
    Single mean0,
    Single mean1,
    Single mean2
)

void PixelVariance(
    EROIBW8 sourceImage,
    EROIBW8 mask,
    Single variance,
    Single mean
)

void PixelVariance(
    EROIBW16 sourceImage,
    EROIBW8 mask,
    Single variance,
    Single mean
)

void PixelVariance(
    EROIC24 sourceImage,
    EROIBW8 mask,
    Single variance0,
    Single variance1,
    Single variance2,
    Single covariance01,
    Single covariance12,
    Single covariance20,
    Single mean0,
    Single mean1,
    Single mean2
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

variance

Reference to the covariances of the pairs of pixel component values.

mean

Reference to the mean pixel component values.

variance0

Reference to the covariances of the pairs of pixel component values.

variance1

Reference to the covariances of the pairs of pixel component values.

variance2

Reference to the covariances of the pairs of pixel component values.

covariance01

Reference to the covariances of the pairs of pixel component values.

covariance12

Reference to the covariances of the pairs of pixel component values.

covariance20

Reference to the covariances of the pairs of pixel component values.

mean0

Reference to the mean pixel component values.

mean1

Reference to the mean pixel component values.

mean2

Reference to the mean pixel component values.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

Remarks

For a color image, computes the means of the three pixel color components, the variances of the components and the covariances between pairs of components.

EasylImage.ProfileDerivative

Computes the first derivative of a profile extracted from a gray-level image.

[VB6]

```

void ProfileDerivative(
    EBW8Vector sourceVector,
    EBW8Vector destinationVector
)

void ProfileDerivative(
    EBW16Vector sourceVector,
    EBW16Vector destinationVector
)

```

Parameters*sourceVector*

Pointer to the source vector.

destinationVector

Pointer to the destination vector.

Remarks

Taking the derivative transforms transitions (edges) into peaks.

Note. Since the **EBW8** data type only handles unsigned values, the derivative is shifted up by 128. Values under [above] 128 correspond to negative [positive] derivative (decreasing [increasing] slope).

EasylImage.ProjectOnAColumn

Projects an image horizontally onto a column.

[VB6]

```

void ProjectOnAColumn(
    EROIBW8 sourceImage,
    EBW32Vector destinationVector
)

void ProjectOnAColumn(
    EROIBW16 sourceImage,
    EBW32Vector destinationVector
)

```

```
void ProjectOnAColumn(
    EROIBW8 sourceImage,
    EBW8Vector destinationVector
)

void ProjectOnAColumn(
    EROIBW16 sourceImage,
    EBW16Vector destinationVector
)

void ProjectOnAColumn(
    EROIC24 sourceImage,
    EC24Vector destinationVector
)

void ProjectOnAColumn(
    EROIBW8 sourceImage,
    EROIBW8 mask,
    EBW32Vector destinationVector
)

void ProjectOnAColumn(
    EROIBW16 sourceImage,
    EROIBW8 mask,
    EBW32Vector destinationVector
)

void ProjectOnAColumn(
    EROIBW8 sourceImage,
    EROIBW8 mask,
    EBW8Vector destinationVector
)

void ProjectOnAColumn(
    EROIBW16 sourceImage,
    EROIBW8 mask,
    EBW16Vector destinationVector
)

void ProjectOnAColumn(
    EROIC24 sourceImage,
    EROIBW8 mask,
    EC24Vector destinationVector
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationVector

Pointer to the destination vector.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

Remarks

Pixel gray/color levels are added when projecting into an [EBW32Vector](#). When projecting into an [EBW8Vector](#)/[EBW16Vector](#)/[EC24Vector](#), pixel values are averaged, instead.

EasylImage.ProjectOnARow

Projects an image vertically onto a row.

[VB6]

```
void ProjectOnARow(
    EROIBW8 sourceImage,
    EBW32Vector destinationVector
)

void ProjectOnARow(
    EROIBW16 sourceImage,
    EBW32Vector destinationVector
)

void ProjectOnARow(
    EROIBW8 sourceImage,
    EBW8Vector destinationVector
)

void ProjectOnARow(
    EROIBW16 sourceImage,
    EBW16Vector destinationVector
)

void ProjectOnARow(
    EROIC24 sourceImage,
    EC24Vector destinationVector
)

void ProjectOnARow(
    EROIBW8 sourceImage,
    EROIBW8 mask,
    EBW32Vector destinationVector
)
```

```
void ProjectOnARow(
    EROIBW16 sourceImage,
    EROIBW8 mask,
    EBW32Vector destinationVector
)

void ProjectOnARow(
    EROIBW8 sourceImage,
    EROIBW8 mask,
    EBW8Vector destinationVector
)

void ProjectOnARow(
    EROIBW16 sourceImage,
    EROIBW8 mask,
    EBW16Vector destinationVector
)

void ProjectOnARow(
    EROIC24 sourceImage,
    EROIBW8 mask,
    EC24Vector destinationVector
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationVector

Pointer to the destination vector.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

Remarks

Pixel gray/color levels are added when projecting into an [EBW32Vector](#). When projecting into an [EBW8Vector](#)/[EBW16Vector](#)/[EC24Vector](#), pixel values are averaged, instead.

EasylImage.RealignFrame

Shifts one frame of the image horizontally.

[VB6]

```

void RealignFrame(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long offset,
    Long fixedRow
)

void RealignFrame(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Long offset,
    Long fixedRow
)

void RealignFrame(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Long offset,
    Long fixedRow
)

```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

offset

Indicates the number of pixels by which to shift (positive to the right).

fixedRow

Specifies which frame remains unchanged (the frame made up of all lines of the same parity as **fixedRow**; by default, **fixedRow = 0**).

Remarks

The same image should be used as source and destination. If the destination image differs from the source image, only the shifted rows are copied. To use a different destination image, the source image must be copied first in the destination image object. When an image is interlaced, the two frames (even and odd lines) are not recorded at the same time. If there is movement in the scene, a visible artifact can result (the edges of objects exhibit a "comb" effect). When the movement is uniform and horizontal (objects on a conveyor belt), one cure to this problem is to shift one of the frames horizontally with respect to the other frame. The amplitude of the shift can be estimated automatically (using [EasylImage::MatchFrames](#)).

EasylImage.RebuildFrame

Rebuilds one frame of the image by interpolation between the lines of the other frame.

[VB6]

```
void RebuildFrame(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long fixedRow
)

void RebuildFrame(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Long fixedRow
)

void RebuildFrame(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Long fixedRow
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

fixedRow

Specifies which frame remains unchanged (the frame made up of all lines of the same parity as **fixedRow**; by default, **fixedRow = 0**).

Remarks

The same image should be used as source and destination. If the destination image differs from the source image, only the shifted rows are copied. To use a different destination image, the source image must be copied first in the destination image object. When an image is interlaced, the two frames (even and odd lines) are not recorded at the same time. If there is movement in the scene, a visible artifact can result (the edges of objects exhibit a "comb" effect). One cure to this problem is to replace one of the frames by linearly interpolating between the lines of the other frame.

EasylImage.RecursiveAverage

Applies stronger noise reduction to small variations and conversely.

[VB6]

```
void RecursiveAverage(
    EROIBW8 sourceImage,
    EROIBW16 store,
    EROIBW8 destinationImage,
    EBW16Vector lookupTable
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

store

Pointer to a 16-bit work image.

destinationImage

Pointer to the destination image/ROI.

lookupTable

Pointer to the LUT vector generated by a call to [EasylImage::SetRecursiveAverageLUT](#).

Remarks

Recursive averaging is a well known process for noise reduction by temporal integration. The principle is to continuously update a noise-free image by blending it, using a linear combination, with the raw, noisy, live image stream. Algorithmically, this amounts to apply the following recurrence: where α is a mixture coefficient. The value of this coefficient can be adjusted so that a prescribed noise reduction ratio is achieved. This procedure is effective when applied to still images, but generates a trailing effect on moving objects because of the transient behavior of the filter. The larger the noise reduction ratio, the heavier the trailing effect. To work around this, a non-linearity can be introduced in the blending process: small gray-level values variations between successive images are usually caused by noise, while large variations correspond to changes in the signal itself (camera displacement or object movements). Function [RecursiveAverage](#) uses this observation and applies stronger noise reduction to small variations and conversely. This way, noise is better reduced in still areas and trailing is avoided in moving areas. For optimal performance, the non-linearity must be pre-computed once for all using function [EasylImage::SetRecursiveAverageLUT](#).

Note. Before the first call to the [RecursiveAverage](#) method, the 16-bit work image *must* be cleared (all pixel values set to zero).

EasylImage.Register

Registers an image by realigning one, two or three pivot points to reference positions.

[VB6]

```
void Register(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Single sourceImagePivot0X,
    Single sourceImagePivot0Y,
    Single destinationImagePivot0X,
    Single destinationImagePivot0Y,
    Long interpolationBits
)
```

```
void Register(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Single sourceImagePivot0X,
    Single sourceImagePivot0Y,
    Single destinationImagePivot0X,
    Single destinationImagePivot0Y,
    Long interpolationBits
)

void Register(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Single sourceImagePivot0X,
    Single sourceImagePivot0Y,
    Single destinationImagePivot0X,
    Single destinationImagePivot0Y,
    Long interpolationBits
)

void Register(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Single sourceImagePivot0X,
    Single sourceImagePivot0Y,
    Single sourceImagePivot1X,
    Single sourceImagePivot1Y,
    Single destinationImagePivot0X,
    Single destinationImagePivot0Y,
    Single destinationImagePivot1X,
    Single destinationImagePivot1Y,
    Long interpolationBits,
    Boolean resize
)
```

```
void Register(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Single sourceImagePivot0X,
    Single sourceImagePivot0Y,
    Single sourceImagePivot1X,
    Single sourceImagePivot1Y,
    Single destinationImagePivot0X,
    Single destinationImagePivot0Y,
    Single destinationImagePivot1X,
    Single destinationImagePivot1Y,
    Long interpolationBits,
    Boolean resize
)

void Register(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Single sourceImagePivot0X,
    Single sourceImagePivot0Y,
    Single sourceImagePivot1X,
    Single sourceImagePivot1Y,
    Single destinationImagePivot0X,
    Single destinationImagePivot0Y,
    Single destinationImagePivot1X,
    Single destinationImagePivot1Y,
    Long interpolationBits,
    Boolean resize
)
```

```
void Register(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Single sourceImagePivot0X,
    Single sourceImagePivot0Y,
    Single sourceImagePivot1X,
    Single sourceImagePivot1Y,
    Single sourceImagePivot2X,
    Single sourceImagePivot2Y,
    Single destinationImagePivot0X,
    Single destinationImagePivot0Y,
    Single destinationImagePivot1X,
    Single destinationImagePivot1Y,
    Single destinationImagePivot2X,
    Single destinationImagePivot2Y,
    Long interpolationBits
)

void Register(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Single sourceImagePivot0X,
    Single sourceImagePivot0Y,
    Single sourceImagePivot1X,
    Single sourceImagePivot1Y,
    Single sourceImagePivot2X,
    Single sourceImagePivot2Y,
    Single destinationImagePivot0X,
    Single destinationImagePivot0Y,
    Single destinationImagePivot1X,
    Single destinationImagePivot1Y,
    Single destinationImagePivot2X,
    Single destinationImagePivot2Y,
    Long interpolationBits
)
```

```
void Register(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Single sourceImagePivot0X,
    Single sourceImagePivot0Y,
    Single sourceImagePivot1X,
    Single sourceImagePivot1Y,
    Single sourceImagePivot2X,
    Single sourceImagePivot2Y,
    Single destinationImagePivot0X,
    Single destinationImagePivot0Y,
    Single destinationImagePivot1X,
    Single destinationImagePivot1Y,
    Single destinationImagePivot2X,
    Single destinationImagePivot2Y,
    Long interpolationBits
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. May not be the same as the source image.

sourceImagePivot0X

First pivot point abscissa in the source image.

sourceImagePivot0Y

First pivot point ordinate in the source image.

destinationImagePivot0X

First pivot point abscissa in the destination image.

destinationImagePivot0Y

First pivot point ordinate in the destination image.

interpolationBits

Number of bits of accuracy for interpolation. Allowed values are **0** (no interpolation, nearest neighbor), **4** or **8**.

sourceImagePivot1X

Second pivot point abscissa in the source image.

sourceImagePivot1Y

Second pivot point ordinate in the source image.

destinationImagePivot1X

Second pivot point abscissa in the destination image.

destinationImagePivot1Y

Second pivot point ordinate in the destination image.

resize

TRUE if scaling is allowed.

sourceImagePivot2X

Third pivot point abscissa in the source image.

sourceImagePivot2Y

Third pivot point ordinate in the source image.

destinationImagePivot2X

Third pivot point abscissa in the destination image.

destinationImagePivot2Y

Third pivot point ordinate in the destination image.

Remarks

Out-of-image-bounds pixels are black. *Registration* is the process of realigning two misaligned images so that point-to-point comparisons are possible. The simplest way to achieve this is to accurately locate features in both images (landmarks or pivots), using pattern matching, point measurement or whatever other technique, and realign one of the images so that the landmarks are superimposed. * When a single pivot point is used, the registration transform is a simple translation. If interpolation bits are used, sub-pixel translation is achieved. * When two pivot points are used, the registration is a combination of translation, rotation and optionally scaling. If scaling is not allowed, the second pivot point will not be matched exactly in general. Anyway, for most applications scaling should not be used unless it corresponds to a change of lens magnification or viewing distance. * When three pivot points are used, the registration is a combination of translation, rotation, shearing correction and optionally scaling. The so-called shear effect can arise when acquiring images with a misaligned line-scan camera. To achieve good accuracy, the pivot points should be chosen as far apart as possible.

EasylImage.RmsNoise

Computes the root-mean-square amplitude of noise, by comparing a given image to a reference image.

[VB6]

```
Single RmsNoise(
    EROIBW8 sourceImage,
    EROIBW8 referenceImage,
    EReferenceNoise referenceNoise
)

Single RmsNoise(
    EROIBW16 sourceImage,
    EROIBW16 referenceImage,
    EReferenceNoise referenceNoise
)

Single RmsNoise(
    EROIC24 sourceImage,
    EROIC24 referenceImage,
    EReferenceNoise referenceNoise
)

Single RmsNoise(
    EROIBW8 sourceImage,
    EROIBW8 referenceImage,
    EROIBW8 mask,
    EReferenceNoise referenceNoise
)

Single RmsNoise(
    EROIBW16 sourceImage,
    EROIBW16 referenceImage,
    EROIBW8 mask,
    EReferenceNoise referenceNoise
)

Single RmsNoise(
    EROIC24 sourceImage,
    EROIC24 referenceImage,
    EROIBW8 mask,
    EReferenceNoise referenceNoise
)

Single RmsNoise(
    EROIBW8 sourceImage,
    EROIBW16 referenceImage,
    Long count,
    EReferenceNoise referenceNoise
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

referenceImage

Pointer to the reference image/ROI.

referenceNoise

Specifies how the reference image is affected by noise, as defined by [EReferenceNoise](#).

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

count

-

Remarks

The reference image can be noiseless (obtained by suppressing the source of noise), or affected by a noise of the same distribution as the given image.

EasylImage.ScaleRotate

Re-scales an image by an arbitrary factor and/or rotates it by an arbitrary angle.

[VB6]

```
void ScaleRotate(
    EROIBW8 sourceImage,
    Single sourceImagePivotX,
    Single sourceImagePivotY,
    Single destinationImagePivotX,
    Single destinationImagePivotY,
    Single scaleX,
    Single scaleY,
    Single rotation,
    EROIBW8 destinationImage,
    Long interpolationBits
)
```

```
void ScaleRotate(
    EROIC24 sourceImage,
    Single sourceImagePivotX,
    Single sourceImagePivotY,
    Single destinationImagePivotX,
    Single destinationImagePivotY,
    Single scaleX,
    Single scaleY,
    Single rotation,
    EROIC24 destinationImage,
    Long interpolationBits
)

void ScaleRotate(
    EROIBW16 sourceImage,
    Single sourceImagePivotX,
    Single sourceImagePivotY,
    Single destinationImagePivotX,
    Single destinationImagePivotY,
    Single scaleX,
    Single scaleY,
    Single rotation,
    EROIBW16 destinationImage,
    Long interpolationBits
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

sourceImagePivotX

Pivot point abscissa in the source image.

sourceImagePivotY

Pivot point ordinate in the source image.

destinationImagePivotX

Pivot point abscissa in the destination image.

destinationImagePivotY

Pivot point ordinate in the destination image.

scaleX

Scale factor for the abscissas.

scaleY

Scale factor for the ordinates.

rotation

Rotation angle, using the current angle unit.

destinationImage

Pointer to the destination image/ROI. May not be the same as the source image.

interpolationBits

Number of bits of accuracy for interpolation. Allowed values are **0** (no interpolation, nearest neighbor), **4** or **8**.

Remarks

For resampling, the nearest neighbor rule or bilinear interpolation with 4 or 8 bits of accuracy is used. The pivot point is a given point in the source image which is mapped to a given point in the destination image. Rotation and scaling are done around the pivot point. Out-of-image-bounds pixels are black.

EasylImage::SetCircleWarp

Prepares suitable warp images for use with function [EasylImage::Warp](#) to un warp a circular ring-wedge shape into a straight rectangle.

[VB6]

```
void SetCircleWarp(
    Single centerX,
    Single centerY,
    Long numberOfRowsRadialSampledPoints,
    Single minimumRadius,
    Single maximumRadius,
    Long numberOfTangentSampledPoints,
    Single minimumAngle,
    Single maximumAngle,
    EIImageBW16 warpImageX,
    EIImageBW16 warpImageY
)
```

Parameters

centerX

Abscissa of the ring-wedge center.

centerY

Ordinate of the ring-wedge center.

numberOfRadialSampledPoints

Number of points to be sampled in the radial direction.

minimumRadius

Starting radius of the ring-wedge shape.

maximumRadius

Ending radius of the ring-wedge shape.

numberOfTangentSampledPoints

Number of points to be sampled in the tangent direction.

minimumAngle

Starting angle of the ring-wedge shape.

maximumAngle

Ending angle of the ring-wedge shape.

warpImageX

Destination warp image for the abscissas.

warpImageY

Destination warp image for the ordinates.

Remarks

Typical use is unwarping of a text printed around a circle.

Note. A ring-wedge is delimited by two concentric circles and two straight lines passing through the center.

EasylImage.SetFrame

Replaces the frame of given parity in an image.

[VB6]

```
void SetFrame(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Boolean odd
)

void SetFrame(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Boolean odd
)
```

```
void SetFrame(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Boolean odd
)
```

Parameters*sourceImage*

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as the source image.

*odd*Specifies which frame is replaced (the frame made up of all lines of the same parity as **odd**).**Remarks**

The size of the destination image is determined as follows: **DstImage_Width = SrcImage_Width**
DstImage_Height = (SrcImage_Height + 1 - odd) / 2

EasylImage.SetRecursiveAverageLUT

Pre-compute the required non-linear transfer function for noise reduction by recursive temporal averaging.

[VB6]

```
void SetRecursiveAverageLUT(
    EBW16Vector lookupTable,
    Single reductionNoiseFactor,
    Single reductionNoiseWidth
)
```

Parameters*lookupTable*

Pointer to the LUT vector holding the non-linear transfer function.

reductionNoiseFactor

Noise reduction factor. The larger the value, the more effectively noise will be reduced.

reductionNoiseWidth

Indicates the extent to which noise reduction applies to large variations in gray-level values. For variations small with respect to this parameter, noise will be reduced by a factor close to the **reductionNoiseFactor** value; for variations much larger than **reductionNoiseWidth**, no noise reduction will take place.

Remarks

This function is a companion to [EasylImage::RecursiveAverage](#).

EasylImage.SetupEqualize

Prepares a lookup-table for image equalization, using an image histogram.

```
[VB6]
void SetupEqualize(
    EBWHistogramVector histogram,
    EBW8Vector lookupTable
)
```

Parameters

histogram

Pointer to the source histogram vector.

lookupTable

Pointer to the destination lookup-table vector.

Remarks

This function, along with [EasylImage::Histogram](#) and [EasylImage::Lut](#), is an alternative to using [EasylImage::Equalize](#).

EasylImage.Shrink

Resizes an image to a smaller size. Pre-filtering is applied to avoid aliasing.

```
[VB6]

void Shrink(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

void Shrink(
    EROIC24 sourceImage,
    EROIC24 destinationImage
)

void Shrink(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

EasylImage.SignalNoiseRatio

Computes the signal to noise ratio, in dB, by comparing a given image to a reference image.

```
[VB6]

Single SignalNoiseRatio(
    EROIBW8 sourceImage,
    EROIBW8 referenceImage,
    EReferenceNoise referenceNoise
)

Single SignalNoiseRatio(
    EROIBW16 sourceImage,
    EROIBW16 referenceImage,
    EReferenceNoise referenceNoise
)
```

```

Single SignalNoiseRatio(
    EROIC24 sourceImage,
    EROIC24 referenceImage,
    EReferenceNoise referenceNoise
)

Single SignalNoiseRatio(
    EROIBW8 sourceImage,
    EROIBW8 referenceImage,
    EROIBW8 mask,
    EReferenceNoise referenceNoise
)

Single SignalNoiseRatio(
    EROIBW16 sourceImage,
    EROIBW16 referenceImage,
    EROIBW8 mask,
    EReferenceNoise referenceNoise
)

Single SignalNoiseRatio(
    EROIC24 sourceImage,
    EROIC24 referenceImage,
    EROIBW8 mask,
    EReferenceNoise referenceNoise
)

Single SignalNoiseRatio(
    EROIBW8 pSrcImage,
    EROIBW16 pRefImage,
    Long un32Count,
    EReferenceNoise eReferenceNoise
)

```

Parameters

sourceImage

Pointer to the source image/ROI.

referenceImage

Pointer to the reference image/ROI.

referenceNoise

Specifies how the reference image is affected by noise, as defined by [EReferenceNoise](#).

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

pSrcImage

```

    - 
    pRefImage
    -
    un32Count
    -
    eReferenceNoise
    -

```

Remarks

The reference image can be noiseless (obtained by suppressing the source of noise) or be affected by a noise of the same distribution as the given image. The signal amplitude is defined as the sum of the squared pixel gray-level values while the noise amplitude is defined as the sum of the squared difference between the pixel gray-level values of the given image and the reference.

EasylImage.SwapFrames

Interchanges the even and odd rows of an image.

[VB6]

```

void SwapFrames(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)

void SwapFrames(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)

void SwapFrames(
    EROIC24 sourceImage,
    EROIC24 destinationImage
)

```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

Remarks

This is helpful when acquisition of an interleaved image has confused even and odd frames.

EasylImage.Thick

Applies a thickening operation on an image, using a 3x3 kernel.

[VB6]

```
void Thick(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    EKernel thickeningKernel,
    EKernelRotation rotationMode,
    Long numberOfIterations
)

void Thick(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    EKernel thickeningKernel,
    EKernelRotation rotationMode,
    Long numberOfIterations
)

void Thick(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    EKernel thickeningKernel,
    EKernelRotation rotationMode,
    Long numberOfIterations
)

Long Thick(
    EROIBW1 sourceImage,
    EROIBW1 destinationImage,
    EKernel thickeningKernel,
    EKernelRotation rotationMode,
    Long numberOfIterations
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as source image.

thickeningKernel

Pointer to the thickening kernel.

rotationMode

Rotation mode, as defined by [EKernelRotation](#).

numberOfIterations

Number of iterations to apply. **0** indicates stop when convergence is reached. Upon return, gives the number of passes actually performed. If the rotation mode is set to either [EKernelRotation_Clockwise](#) or [EKernelRotation_Anticlockwise](#), a pass comprises eight kernel rotations.

Remarks

The thickening kernel coefficients must be **0** (matching black pixel, value 0), **1** (matching non black pixel, value > 0) or **-1** (don't care). When a match is found between the kernel coefficients and the neighborhood of a pixel, the pixel value is set to **255**.

Easylimage.Thin

Applies a thinning operation on an image, using a 3x3 kernel.

[VB6]

```
void Thin(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    EKernel thinningKernel,
    EKernelRotation rotationMode,
    Long numberOfIterations
)

void Thin(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    EKernel thinningKernel,
    EKernelRotation rotationMode,
    Long numberOfIterations
)
```

```

void Thin(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    EKernel thinningKernel,
    EKernelRotation rotationMode,
    Long numberOfIterations
)

Long Thin(
    EROIBW1 sourceImage,
    EROIBW1 destinationImage,
    EKernel thinningKernel,
    EKernelRotation rotationMode,
    Long numberOfIterations
)

```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as source image.

thinningKernel

Pointer to the thinning kernel.

rotationMode

Rotation mode, as defined by [EKernelRotation](#).

numberOfIterations

Number of iterations to apply. **0** indicates stop when convergence is reached. Upon return, gives the number of passes actually performed. If the rotation mode is set to either [EKernelRotation_Clockwise](#) or [EKernelRotation_Anticlockwise](#), a pass comprises eight kernel rotations.

Remarks

The thinning kernel coefficients must be **0** (matching black pixel, value 0), **1** (matching non black pixel, value > 0) or **-1** (don't care). When a match is found between the kernel coefficients and the neighborhood of a pixel, the pixel value is set to 0.

EasylImage.ThreeLevelsMinResidueThreshold

Computes the two threshold values used to separate the pixels of an image in three classes.

[VB6]

```
Single ThreeLevelsMinResidueThreshold(
    EBWHistogramVector histogram,
    EBW8 firstGrayPixelValue,
    EBW8 firstWhitePixelValue,
    Single averageBlack,
    Single averageGray,
    Single averageWhite
)
```

Parameters

histogram

Histogram of the image.

firstGrayPixelValue

Low threshold.

firstWhitePixelValue

High threshold.

averageBlack

Average value of the black pixels (pixels under the low threshold).

averageGray

Average value of the gray pixels (pixels between the low threshold and the high threshold).

averageWhite

Average value of the white pixels (pixels over the high threshold).

Remarks

These values are computed using the minimum residue criterion from the histogram of the image. The function returns the minimum residue as per the method. The residue is the Euclidian distance between the source image and the thresholded image.

EasylImage.Threshold

Binarize an image by setting pixels to two different possible values in a destination image, according to their value in a source image.

[VB6]

```
void Threshold(
    EROIBW8 sourceImage,
    EROIBW1 destinationImage,
    Long threshold,
    Single relativeThreshold
)

void Threshold(
    EROIBW16 sourceImage,
    EROIBW1 destinationImage,
    Long threshold,
    Single relativeThreshold
)

void Threshold(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long threshold,
    Byte lowValue,
    Byte highValue,
    Single relativeThreshold
)

void Threshold(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage
)

void Threshold(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Long threshold
)

void Threshold(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Long threshold,
    EBW16 lowValue,
    EBW16 highValue
)

void Threshold(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Single relativeThreshold
)
```

```

void Threshold(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Single relativeThreshold,
    EBW16 lowValue,
    EBW16 highValue
)

void Threshold(
    EROIC24 sourceImage,
    EROIBW8 destinationImage,
    EC24 minimum,
    EC24 maximum
)

void Threshold(
    EROIC24 sourceImage,
    EROIBW8 destinationImage,
    EC24 minimum,
    EC24 maximum,
    EColorLookup colorLookupTable,
    EBW8 rejectValue,
    EBW8 acceptValue
)

void Threshold(
    EROIC24 sourceImage,
    EROIBW8 destinationImage,
    EC24 minimum,
    EC24 maximum,
    EColorLookup colorLookupTable
)

```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

threshold

The value to compare each pixel to

relativeThreshold

Fraction of the image pixels that will be set below the threshold. Only used when the threshold value is [EThresholdMode_Relative](#) (by default, **0.5**). This value must be greater than (or equal to) 0 and strictly less than 1.

lowValue

Value for pixels below the threshold (by default, **0**).

highValue

Value for pixels above the threshold (by default, it is set to **255** for BW8 destination images and **65535** for BW16 destination images).

minimum

Three lower thresholds combined in a single color value.

maximum

Three upper thresholds combined in a single color value.

colorLookupTable

Pointer to the color lookup table to be applied before thresholding, if any.

rejectValue

Value for pixels falling outside the range (by default, **0**).

acceptValue

Value for pixels falling inside the range (by default, **255**).

Remarks

When the source image is gray-level, the pixel values are measured against a threshold. All pixels below this threshold will yield a low value in the destination image, and all pixels above or on the threshold will yield a high value. When the destination image is binary (BW1 pixel type), then the values are set to **0** or **1**, according to the criterion. When the destination image is gray-level (BW8 or BW16), then the values are set to **0** or to the maximum pixel value for the image type (**255** for BW8 and **65535** for BW16). In some overloads, these minimum and maximum destination values can be specified. When the source image is gray-level, several modes are available: absolute (the threshold value is given), relative (the threshold value is computed to obtain a desired fraction of the image pixels), or automatic (using three different criteria). In the function overloads where this mode cannot be specified, it is assumed to be absolute. If the source image is color, all pixels whose components are comprised in a range of values (minimum to maximum) will be set to a constant value (white by default), while other pixels will be set to another constant value (black by default). In this case, if a color lookup is specified, it is applied on the fly to the color image before thresholding. The simpler color image overload does not support the use of an on-the-fly color lookup table nor **rejectValue/acceptValue** arguments. On the other hand, it has been MMX optimized, and will run significantly faster when the acceptance region is large.

EasylImage.TwoLevelsMinResidueThreshold

Computes the threshold value used to separate the pixels of an image in two classes.

[VB6]

```
Single TwoLevelsMinResidueThreshold(
    EBWHistogramVector histogram,
    EBW8 firstWhitePixelValue,
    Single averageBlack,
    Single averageWhite
)
```

Parameters

histogram

Histogram of the image.

firstWhitePixelValue

Threshold.

averageBlack

Average value of the black pixels (pixels under the threshold).

averageWhite

Average value of the white pixels (pixels over the threshold).

Remarks

This value is computed using the minimum residue criterion from the histogram of the image. The function returns the minimum residue as per the method. The residue is the Euclidian distance between the source image and the thresholded image.

EasylImage.Uniformize

Shading correction is the process of transforming the gray or color component values of an image, using one or two reference images or vectors.

[VB6]

```
void Uniformize(
    EROIBW8 sourceImage,
    EBW8 pixelReference,
    EROIBW8 imageReference,
    EROIBW8 destinationImage,
    Boolean multiplicative
)
```

```
void Uniformize(
    EROIBW16 sourceImage,
    EBW16 pixelReference,
    EROIBW16 imageReference,
    EROIBW16 destinationImage,
    Boolean multiplicative
)

void Uniformize(
    EROIC24 sourceImage,
    EC24 pixelReference,
    EROIC24 imageReference,
    EROIC24 destinationImage,
    Boolean multiplicative
)

void Uniformize(
    EROIBW8 sourceImage,
    EBW8 pixelReference,
    EBW8Vector vectorOfPixelReference,
    EROIBW8 destinationImage,
    Boolean multiplicative
)

void Uniformize(
    EROIBW16 sourceImage,
    EBW16 pixelReference,
    EBW16Vector vectorOfPixelReference,
    EROIBW16 destinationImage,
    Boolean multiplicative
)

void Uniformize(
    EROIC24 sourceImage,
    EC24 pixelReference,
    EC24Vector vectorOfPixelReference,
    EROIC24 destinationImage,
    Boolean multiplicative
)

void Uniformize(
    EROIBW8 sourceImage,
    EBW8 pixelLightReference,
    EROIBW8 imageLightReference,
    EBW8 pixelDarkReference,
    EROIBW8 imageDarkReference,
    EROIBW8 destinationImage
)
```

```
void Uniformize(
    EROIBW16 sourceImage,
    EBW16 pixelLightReference,
    EROIBW16 imageLightReference,
    EBW16 pixelDarkReference,
    EROIBW16 imageDarkReference,
    EROIBW16 destinationImage
)

void Uniformize(
    EROIC24 sourceImage,
    EC24 pixelLightReference,
    EROIC24 imageLightReference,
    EC24 pixelDarkReference,
    EROIC24 imageDarkReference,
    EROIC24 destinationImage
)

void Uniformize(
    EROIBW8 sourceImage,
    EBW8 pixelLightReference,
    EBW8Vector vectorOfPixelLightReference,
    EBW8 pixelDarkReference,
    EBW8Vector vectorOfPixelDarkReference,
    EROIBW8 destinationImage
)

void Uniformize(
    EROIBW16 sourceImage,
    EBW16 pixelLightReference,
    EBW16Vector vectorOfPixelLightReference,
    EBW16 pixelDarkReference,
    EBW16Vector vectorOfPixelDarkReference,
    EROIBW16 destinationImage
)

void Uniformize(
    EROIC24 sourceImage,
    EC24 pixelLightReference,
    EC24Vector vectorOfPixelLightReference,
    EC24 pixelDarkReference,
    EC24Vector vectorOfPixelDarkReference,
    EROIC24 destinationImage
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

pixelReference

Constant value to transform the reference image or vector into.

imageReference

Pointer to the reference source image/ROI or vector.

destinationImage

Pointer to the destination image/ROI.

multiplicative

TRUE, if the transform is multiplicative (gain); **FALSE**, if the transform is additive (offset) (by default, **TRUE**).

vectorOfPixelReference

Constant value to transform the reference image or vector into.

pixelLightReference

Constant value to transform the light reference image or vector into.

imageLightReference

Pointer to the light reference source image/ROI or vector.

pixelDarkReference

Constant value to transform the dark reference image/ROI or vector into.

imageDarkReference

Pointer to the dark reference source image/ROI or vector.

vectorOfPixelLightReference

Constant value to transform the light reference image or vector into.

vectorOfPixelDarkReference

Constant value to transform the dark reference image/ROI or vector into.

Remarks

The intent is to compensate for non-uniform lighting or sensor response non-uniformity by providing images of the background with no foreground object present. In the case of area-scan cameras, the illumination can change anywhere in the field of view, requiring 2D compensation. In the case of line-scan cameras imaging moving parts, illumination remains constant across image rows. Only 1D compensation is required. In this case, the reference illumination is specified as a vector, which is replicated across all image rows. * When a single reference image is used, the transform is analog to an adaptive (space-variant) gain or offset (Gain * Intensity or Intensity + Offset); the transform lets the reference image(s) become a specified constant value, i.e. flat field illumination. * When two reference images are used, the transform is analog to adaptive gain *and* offset (Gain * Intensity + Offset); the transform let both reference images become specified constants, i.e. flat field illumination with a correct black reference.

Note. The reference image(s) should be chosen such that they contain no saturated pixel values (remain in the linear domain) and little (filtered out) noise.

EasylImage.VerticalMirror

Mirrors an image vertically (the rows are swapped).

```
[VB6]

void VerticalMirror(
    EROIBW8 sourceImage
)

void VerticalMirror(
    EROIBW16 sourceImage
)

void VerticalMirror(
    EROIC24 sourceImage
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

EasylImage.Warp

Transforms an image so that each pixels are moved to the locations specified in the "warp" images used as look-up tables.

```
[VB6]

void Warp(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    EImageBW16 warpImageX,
    EImageBW16 warpImageY,
    Long shiftX,
    Long shiftY
)
```

```
void Warp(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    EIImageBW16 warpImageX,
    EIImageBW16 warpImageY,
    Long shiftX,
    Long shiftY
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

warpImageX

Pointer to the X lookup image.

warpImageY

Pointer to the Y lookup image.

shiftX

Horizontal translation.

shiftY

Vertical translation.

Remarks

For example, pixel [10,20] moves to location [WarpXImage[10,20], WarpYImage[10,20]].

EasylImage.WeightedMoments

Computes the zero-th, first, second, third or fourth order weighted moments on the gray-level image. The weight of a pixel is its gray-level value.

[VB6]

```
void WeightedMoments (
    EROIBW8 sourceImage,
    Single M,
    Single Mx,
    Single My
)

void WeightedMoments (
    EROIBW16 sourceImage,
    Single M,
    Single Mx,
    Single My
)

void WeightedMoments (
    EROIBW8 sourceImage,
    Single M,
    Single Mx,
    Single My,
    Single Mxx,
    Single Mxy,
    Single Myy
)

void WeightedMoments (
    EROIBW16 sourceImage,
    Single M,
    Single Mx,
    Single My,
    Single Mxx,
    Single Mxy,
    Single Myy
)

void WeightedMoments (
    EROIBW8 sourceImage,
    Single M,
    Single Mx,
    Single My,
    Single Mxx,
    Single Mxy,
    Single Myy,
    Single Mxxx,
    Single Mxxy,
    Single Mxyy,
    Single Myyy
)
```

```
void WeightedMoments (
    EROIBW16 sourceImage,
    Single M,
    Single Mx,
    Single My,
    Single Mxx,
    Single Mxy,
    Single Myy,
    Single Mxxx,
    Single Mxxy,
    Single Mxyy,
    Single Myyy
)

void WeightedMoments (
    EROIBW8 sourceImage,
    Single M,
    Single Mx,
    Single My,
    Single Mxx,
    Single Mxy,
    Single Myy,
    Single Mxxx,
    Single Mxxy,
    Single Mxyy,
    Single Myyy,
    Single Mxxxx,
    Single Mxxxxy,
    Single Mxxxxy,
    Single Mxyyy,
    Single Myyyy
)
```

```
void WeightedMoments (
    EROIBW16 sourceImage,
    Single M,
    Single Mx,
    Single My,
    Single Mxx,
    Single Mxy,
    Single Myy,
    Single Mxxx,
    Single Mxxy,
    Single Mxyy,
    Single Myyy,
    Single Mxxxx,
    Single Mxxxxy,
    Single Mxxyy,
    Single Mxyyy,
    Single Myyyy
)

void WeightedMoments (
    EROIBW8 sourceImage,
    EROIBW8 mask,
    Single M,
    Single Mx,
    Single My
)

void WeightedMoments (
    EROIBW16 sourceImage,
    EROIBW8 mask,
    Single M,
    Single Mx,
    Single My
)

void WeightedMoments (
    EROIBW8 sourceImage,
    EROIBW8 mask,
    Single M,
    Single Mx,
    Single My,
    Single Mxx,
    Single Mxy,
    Single Myy
)
```

```
void WeightedMoments (
    EROIBW16 sourceImage,
    EROIBW8 mask,
    Single M,
    Single Mx,
    Single My,
    Single Mxx,
    Single Mxy,
    Single Myy
)

void WeightedMoments (
    EROIBW8 sourceImage,
    EROIBW8 mask,
    Single M,
    Single Mx,
    Single My,
    Single Mxx,
    Single Mxy,
    Single Myy,
    Single Mxxx,
    Single Mxxy,
    Single Mxyy,
    Single Myyy
)

void WeightedMoments (
    EROIBW16 sourceImage,
    EROIBW8 mask,
    Single M,
    Single Mx,
    Single My,
    Single Mxx,
    Single Mxy,
    Single Myy,
    Single Mxxx,
    Single Mxxy,
    Single Mxyy,
    Single Myyy
)
```

```
void WeightedMoments (
    EROIBW8 sourceImage,
    EROIBW8 mask,
    Single M,
    Single Mx,
    Single My,
    Single Mxx,
    Single Mxy,
    Single Myy,
    Single Mxxx,
    Single Mxxy,
    Single Mxyy,
    Single Myyy,
    Single Mxxxx,
    Single Mxxxxy,
    Single Mxxyy,
    Single Mxyyy,
    Single Myyyy
)

void WeightedMoments (
    EROIBW16 sourceImage,
    EROIBW8 mask,
    Single M,
    Single Mx,
    Single My,
    Single Mxx,
    Single Mxy,
    Single Myy,
    Single Mxxx,
    Single Mxxy,
    Single Mxyy,
    Single Myyy,
    Single Mxxxx,
    Single Mxxxxy,
    Single Mxxyy,
    Single Mxyyy,
    Single Myyyy
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

M

Reference to the zero-th order weighted moment (total gray value).

Mx

Reference to the first order moments (weighted sums of abscissas and ordinates).

My

Reference to the first order moments (weighted sums of abscissas and ordinates).

Mxx

Reference to the second order uncentered moments (weighted sums of squared abscissas, cross-product and squared ordinates).

Mxy

Reference to the second order uncentered moments (weighted sums of squared abscissas, cross-product and squared ordinates).

Myy

Reference to the second order uncentered moments (weighted sums of squared abscissas, cross-product and squared ordinates).

Mxxx

Reference to the third order uncentered moments (weighted sums of third order products).

Mxxy

Reference to the third order uncentered moments (weighted sums of third order products).

Mxyy

Reference to the third order uncentered moments (weighted sums of third order products).

Myyy

Reference to the third order uncentered moments (weighted sums of third order products).

Mxxxx

Reference to the fourth order uncentered moments (weighted sums of fourth order products).

Mxxxxy

Reference to the fourth order uncentered moments (weighted sums of fourth order products).

Mxxyy

Reference to the fourth order uncentered moments (weighted sums of fourth order products).

Mxyyy

Reference to the fourth order uncentered moments (weighted sums of fourth order products).

Myyyy

Reference to the fourth order uncentered moments (weighted sums of fourth order products).

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

EasylImage.WhiteTopHatBox

Performs a top-hat filtering on an image (source image minus opened image) on a rectangular kernel.

[VB6]

```
void WhiteTopHatBox(
    EROIBW1 sourceImage,
    EROIBW1 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

void WhiteTopHatBox(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

void WhiteTopHatBox(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)

void WhiteTopHatBox(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Long halfOfKernelWidth,
    Long halfOfKernelHeight
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as the source image.

halfOfKernelWidth

Half of the box width minus one (by default, **halfOfKernelWidth = 1**; **0** is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as **halfOfKernelWidth**; **0** is allowed).

Remarks

This filter enhances the thin white features.

EasylImage.WhiteTopHatDisk

Performs a top-hat filtering on an image (source image minus opened image) on a quasi-circular kernel.

```
[VB6]
void WhiteTopHatDisk(
    EROIBW1 sourceImage,
    EROIBW1 destinationImage,
    Long halfOfKernelWidth
)

void WhiteTopHatDisk(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Long halfOfKernelWidth
)

void WhiteTopHatDisk(
    EROIBW16 sourceImage,
    EROIBW16 destinationImage,
    Long halfOfKernelWidth
)

void WhiteTopHatDisk(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Long halfOfKernelWidth
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as the source image.

halfOfKernelWidth

Half width of the kernel minus one (by default, **halfOfKernelWidth = 1**; **0** is allowed).

Remarks

This filter enhances the thin white features.

EasyObject Class

This class contains static properties and methods specific to the EasyObject library.

METHODS

ContourArea

Computes the area of an object from its contour.

ContourGravityCenter

Computes the area and gravity center of an object from its contour.

ContourInertia

Computes the inertia parameters of an object from its contour.

IsFloatFeature

Tests whether a given feature is associated with floating-point values.

IsIntegerFeature

Tests whether a given feature is associated with integer values.

IsUnsignedIntegerFeature

Tests whether a given feature is associated with unsigned integer values.

EasyObject.ContourArea

Computes the area of an object from its contour.

```
[VB6]  
void ContourArea(  
    EPathVector pPathVector,  
    Long n32Area  
)
```

Parameters

pPathVector

Pointer to the source vector.

n32Area

Reference to the area to compute.

EasyObject.ContourGravityCenter

Computes the area and gravity center of an object from its contour.

```
[VB6]  
void ContourGravityCenter(  
    EPathVector pPathVector,  
    Long n32Area,  
    Single f32GravityCenterX,  
    Single f32GravityCenterY  
)
```

Parameters

pPathVector

Pointer to the source vector.

n32Area

Reference to the area to compute.

f32GravityCenterX

Reference to the abscissa of the gravity center to compute.

f32GravityCenterY

Reference to the ordinate of the gravity center to compute.

EasyObject.ContourInertia

Computes the inertia parameters of an object from its contour.

[VB6]

```
void ContourInertia(
    EPathVector pPathVector,
    Long n32Area,
    Single f32GravityCenterX,
    Single f32GravityCenterY,
    Single f32SigmaX,
    Single f32SigmaY,
    Single f32SigmaXY
)
```

Parameters

pPathVector

Pointer to the source vector.

n32Area

Reference to the area to compute.

f32GravityCenterX

Reference to the abscissa of the gravity center to compute.

f32GravityCenterY

Reference to the ordinate of the gravity center to compute.

f32SigmaX

Centered cross moment of inertia.

f32SigmaY

Centered moment of inertia around Y.

f32SigmaXY

Centered cross moment of inertia.

EasyObject::IsFloatFeature

Tests whether a given feature is associated with floating-point values.

[VB6]

```
Boolean IsFloatFeature(  
    EFeature feature  
)
```

Parameters

feature

The feature.

Remarks

Most features are floating-point. The exceptions are listed in these functions:

[EasyObject::IsUnsignedIntegerFeature](#) and [EasyObject::IsIntegerFeature](#).

EasyObject::IsIntegerFeature

Tests whether a given feature is associated with integer values.

[VB6]

```
Boolean IsIntegerFeature(  
    EFeature feature  
)
```

Parameters

feature

The feature.

Remarks

The features associated with integer values are: [EFeature_ContourX](#), [EFeature_ContourY](#), [EFeature_LeftLimit](#), [EFeature_RightLimit](#), [EFeature_TopLimit](#), and [EFeature_BottomLimit](#).

EasyObject.IsUnsignedIntegerFeature

Tests whether a given feature is associated with unsigned integer values.

[VB6]

```
Boolean IsUnsignedIntegerFeature(  
    EFeature feature  
)
```

Parameters

feature

The feature.

Remarks

The features associated with unsigned integer values are: [EFeature_ElementIndex](#), [EFeature_LayerIndex](#), [EFeature_RunCount](#), [EFeature_Area](#) and [EFeature_LargestRun](#).

EBarCode Class

Manages a complete context for the reading or verification of bar codes in EasyBarCode.

Base Class: [ERectangleShape](#)

PROPERTIES

AdditionalSymbolologies

Enabled symbologies belonging to the group of additional symbologies.

KnownLocation

Flag indicating whether the symbol location is known or not.

KnownModule	Flag indicating whether the symbol module is known or not.
Module	Module value.
NumDecodedSymbologies	Number of symbologies (among the enabled ones) for which the decoding process was successful.
NumEnabledSymbologies	Number of enabled symbologies.
Rectangle	Sets the nominal position (center coordinates), size and rotation angle of the symbol bounding box according to a known rectangle.
RelativeReadingSizeX	Reading area width, relative to the symbol extent.
RelativeReadingSizeY	Reading area height, relative to the symbol extent.
RelativeReadingX	Reading area abscissa, relative to the symbol position.
RelativeReadingY	Reading area ordinate, relative to the symbol position.
StandardSymbologies	Enabled symbologies belonging to the group of standard symbologies.

ThicknessRatio	Bars thickness ratio.
VerifyChecksum	M "VerifyChecksum" mode. E
THODS	
Decode	Provides the decoded information (or a reading error code) corresponding to the specified symbology.
Detect	Processes the image, reads the possible contents of the bar code corresponding to all enabled symbologies and rates their likeliness.
Drag	Moves a handle to a new position and updates the position parameters of the symbol bounding box.
Draw	Draws the symbol bounding box.
DrawWithCurrentPen	Draws the symbol bounding box.
EBarcode	Creates an EBarcode object.

GetDecodedAngle	Allows retrieving the bar code reading angle, pertaining to the specified symbology, or, at least, to the most likely symbology.
GetDecodedDirection	Returns the encoding direction of the bar code, pertaining to the specified symbology, or, at least, to the most likely symbology.
GetDecodedRectangle	Allows retrieving the bar code reading rectangle, pertaining to the specified symbology, or, at least, to the most likely symbology.
GetDecodedSymbology	Returns the identifier of one of the symbologies that were successfully decoded.
GetSymbolName	-
HitTest	Checks whether the cursor is positioned over a handle (TRUE) or not (FALSE).
Read	Processes the image, reads the possible contents of the bar code and returns the single possible decoding (mono-symbology) or the most likely one (multi-symbology) or a reading error code.
SetReadingCenter	Sets the reading area center coordinates, relative to the symbol position and extent.

SetReadingSize

ESets the reading area size, relative to the symbol extent.

B

arCode.AdditionalSymbologies

Enabled symbologies belonging to the group of additional symbologies.

[VB6]

AdditionalSymbologies As Long

read-write

Remarks

Due to the large number of supported symbologies, they have been gathered in two groups. For more information about these groups, see [ESymbologies](#).

EBarCode.Decode

Provides the decoded information (or a reading error code) corresponding to the specified symbology.

[VB6]

```
String Decode(
    ESymbologies symbology
)
```

Parameters

symbology

Specified symbology, as defined by [ESymbologies](#) this symbology must have been enabled).

Remarks

Before calling [EBarCode::Decode](#), an [EBarCode::Detect](#) operation must have been performed. In case of the mono-symbology mode, or if only the most likely decoding matters, the [EBarCode::Read](#) method should be used.

EBarCode.Detect

Processes the image, reads the possible contents of the bar code corresponding to all enabled symbologies and rates their likeliness.

[VB6]

```
void Detect(
    EROIBW8 sourceImage
)
```

Parameters

sourceImage

Pointer to the image containing the bar code.

Remarks

Processing the image means finding the symbol (in case of automatic location mode) and retrieving its descriptive parameters. The decoded information corresponding to a specific symbology is provided by the decode function. The symbologies that were successfully decoded are ranked by decreasing likeliness (range **0** to **NumDecodedSymbologies-1**). In case of the mono-symbology mode or if only the most likely decoding matters, the [Read](#) function should be used.

EBarCode.Drag

Moves a handle to a new position and updates the position parameters of the symbol bounding box.

[VB6]

```
void Drag(
    Long cursorX,
    Long cursorY
)
```

Parameters

cursorX
Cursor current coordinates.
cursorY
Cursor current coordinates.

EBarCode.Draw

Draws the symbol bounding box.

[VB6]

```
void Draw(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)

void Draw(
    EDrawAdapter graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)

void Draw(
    Long graphicContext,
    ERGBColor color,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

Parameters

graphicContext
Handle of the device context on which to draw.
drawingMode

Indicates how the symbol bounding box must be displayed, as defined by [EDrawingMode](#).
daughters

TRUE if the shapes attached to the symbol bounding box are to be displayed as well.

color

The color in which to draw the overlay.

Remarks

The bounding box corresponds to the nominal position of the bar code ([EDrawingMode_Nominal](#)), in case this information has been explicitly provided, and to the actual position ([EDrawingMode_Actual](#)) if it has been determined by image analysis.

EBarCode.DrawWithCurrentPen

Draws the symbol bounding box.

[VB6]

```
void DrawWithCurrentPen(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the symbol bounding box must be displayed, as defined by [EDrawingMode](#).
daughters

TRUE if the shapes attached to the symbol bounding box are to be displayed as well.

Remarks

The bounding box corresponds to the nominal position of the bar code ([EDrawingMode_Nominal](#)), in case this information has been explicitly provided, and to the actual position ([EDrawingMode_Actual](#)) if it has been determined by image analysis.

EBarCode.EBarCode

Creates an [EBarCode](#) object.

```
[VB6]  
void EBarCode(  
    EBarCode other  
)  
  
void EBarCode(  
)
```

Parameters

other

-

EBarCode.GetDecodedAngle

Allows retrieving the bar code reading angle, pertaining to the specified symbology, or, at least, to the most likely symbology.

```
[VB6]  
void GetDecodedAngle(  
    Single decodedAngle  
)  
  
void GetDecodedAngle(  
    Single decodedAngle,  
    Single cutAngle  
)  
  
void GetDecodedAngle(  
    ESymbologies symbology,  
    Single decodedAngle  
)
```

```
void GetDecodedAngle(
    ESymbologies symbology,
    Single decodedAngle,
    Single cutAngle
)
```

Parameters*decodedAngle*

Returned bar code reading angle value.

cutAngle

Cut angle value (Â°) defining the allowed range for the bar code reading angle ([**cutAngle**, **cutAngle + 360**]). By default, the cut angle equals **-45**.

symbology

Specified symbology, as defined by [ESymbologies](#) (this symbology must have been enabled).

EBarCode.GetDecodedDirection

Returns the encoding direction of the bar code, pertaining to the specified symbology, or, at least, to the most likely symbology.

[VB6]

```
void GetDecodedDirection(
    Boolean directEncoding
)

void GetDecodedDirection(
    ESymbologies symbology,
    Boolean directEncoding
)
```

Parameters*directEncoding*

Boolean holding the encoding direction status.

symbology

Specified symbology, as defined by [ESymbologies](#) (this symbology must have been enabled).

Remarks

The encoding direction of the bar code is "Direct" or "Inverse". The encoding direction is said to be "Direct" when the bar code longitudinal axis falls in the range [-45°, 135]. Conversely, when the bar code longitudinal axis doesn't fall in the previous range, the encoding direction is said to be "Inverse".

EBarCode.GetDecodedRectangle

Allows retrieving the bar code reading rectangle, pertaining to the specified symbology, or, at least, to the most likely symbology.

```
[VB6]  
  
void GetDecodedRectangle(  
    ERectangle rect  
)  
  
void GetDecodedRectangle(  
    ESymbologies symbology,  
    ERectangle rect  
)
```

Parameters

rect

Returned bar code reading rectangle.

symbology

Specified symbology, as defined by [ESymbologies](#) (this symbology must have been enabled).

EBarCode.GetDecodedSymbology

Returns the identifier of one of the symbologies that were successfully decoded.

```
[VB6]
```

```
ESymbolologies GetDecodedSymbolology(
```

Parameters

index

Index of the specified symbology (range **0** to **NumDecodedSymbolologies-1**).

Remarks

The desired symbology is specified by its ranking index (range **0** to **NumDecodedSymbologies-1**). The symbologies that were successfully decoded are ranked by decreasing likeliness.

EBarCode.GetSymbologyName

—

```
String GetSymbolName(
```

Parameters

symbology

EBarCode.HitTest

Checks whether the cursor is positioned over a handle (**TRUE**) or not (**FALSE**).

[VB6]

```
Boolean HitTest(  
    Boolean daughters  
)
```

Parameters

daughters

TRUE if the handles of the shapes attached to the symbol bounding box have to be considered as well.

EBarCode.KnownLocation

Flag indicating whether the symbol location is known or not.

[VB6]

```
KnownLocation As Boolean  
read-write
```

Remarks

In case of known location, use [EBarCode::Rectangle](#) to adjust a rectangle around the symbol.

EBarCode.KnownModule

Flag indicating whether the symbol module is known or not.

[VB6]

```
KnownModule As Boolean  
read-write
```

Remarks

If **TRUE**, it is also necessary to get the **EBarCode::Module** and **EBarCode::ThicknessRatio** properties in order to specify the requested module and thickness ratio.

EBarCode.Module

Module value.

[VB6]

Module As Single

read-write

Remarks

The module value is a descriptive parameter participating in the encoding; it corresponds to the thinner bar width. Symbols whose bars thickness can take two values, the module and **V** times the module (where **V** runs from **1.5** to **3**), are called *binary* bar codes. When the bars thickness are small integer multiples (1 to 4 or 5) of a module, the symbols are called *modular* bar codes.

EBarCode.NumDecodedSymbologies

Number of symbologies (among the enabled ones) for which the decoding process was successful.

[VB6]

NumDecodedSymbologies As Long

read-only

EBarCode.NumEnabledSymbologies

Number of enabled symbologies.

[VB6]

NumEnabledSymbologies As Long

read-only

EBarCode.Read

Processes the image, reads the possible contents of the bar code and returns the single possible decoding (mono-symbology) or the most likely one (multi-symbology) or a reading error code.

[VB6]

```
String Read(  
    EROIBW8 sourceImage  
)
```

Parameters

sourceImage

The image containing the bar code.

Remarks

Processing the image means finding the symbol (in case of automatic location mode) and retrieving its descriptive parameters. When decoding other than the most likely one are also required, a call to the [EBarCode::Detect](#) function followed by a [EBarCode::Decode](#) should be used.

EBarCode.Rectangle

Sets the nominal position (center coordinates), size and rotation angle of the symbol bounding box according to a known rectangle.

[VB6]

Rectangle As ERectangle

read-write

Remarks

An [ERectangle](#) object is characterized by its center coordinates, its size and its rotation angle.

EBarCode.RelativeReadingSizeX

Reading area width, relative to the symbol extent.

[VB6]

RelativeReadingSizeX As Single

read-only

EBarCode.RelativeReadingSizeY

Reading area height, relative to the symbol extent.

[VB6]

RelativeReadingSizeY As Single

read-only

EBarCode.RelativeReadingX

Reading area abscissa, relative to the symbol position.

[VB6]

RelativeReadingX As Single

read-only

EBarCode.RelativeReadingY

Reading area ordinate, relative to the symbol position.

[VB6]

RelativeReadingY As Single

read-only

EBarCode.SetReadingCenter

Sets the reading area center coordinates, relative to the symbol position and extent.

[VB6]

```
void SetReadingCenter(
    Single relativeX,
    Single relativeY
)
```

Parameters

relativeX

Reading area center abscissa, relative to the symbol position and extent.

relativeY

Reading area center ordinate, relative to the symbol position and extent.

EBarCode.SetReadingSize

Sets the reading area size, relative to the symbol extent.

[VB6]

```
void SetReadingSize(
    Single relativeSizeX,
    Single relativeSizeY
)
```

Parameters

relativeSizeX

Reading area width, relative to the symbol extent.

relativeSizeY

Reading area height, relative to the symbol extent.

EBarCode.StandardSymbologies

Enabled symbologies belonging to the group of standard symbologies.

[VB6]

StandardSymbolologies As Long

read-write

Remarks

Due to the large number of supported symbologies, they have been gathered in two groups.
For more information about these groups, see [ESymbologies](#).

EBarCode.ThicknessRatio

Bars thickness ratio.

[VB6]

ThicknessRatio As Single

read-write

Remarks

This property is relevant in case of binary codes only. It corresponds to the ratio of a thin bar width over a thick bar width, and should range from **1.5** to **3**.

EBarCode.VerifyChecksum

"VerifyChecksum" mode.

[VB6]

VerifyChecksum As Boolean

read-write

Remarks

The "VerifyChecksum" mode enables or disables verification of the checksum character. That verification mode is set in the same way for all enabled symbologies. It is worth noting that checksum may be present or not in the bar code, and the user may verify or not checksum validity. These two circumstances are independent, and give rise to four modes of operation. The verification process will return an "invalid checksum" error in case of bad checksum character. This error can also be generated if there is no checksum in the bar code. In the other case, when checksum are not verified, no error will occur, and the process will continue silently. When the "VerifyChecksum" mode is enabled, the returned decoded string does not contain the checksum character(s). Conversely, when the verification process is disabled, the checksum character(s) are concatenated to the encoded information.

EBaseROI Class

This represents the abstract base class for all ROI and image classes.

Derived Class(es): [EROIBW8](#) [EROIBW16](#) [EROIBW32](#) [EROIC24](#) [EROIBW1](#) [EROIC15](#) [EROIC16](#) [EROIC24A](#) [EROIC48](#)

PROPERTIES

Author

Gets or sets the Author attribute of the ROI/image. This is a convenience property that allows storing some information in the ROI/image. Please note that not all file formats preserve this information.

BaseTopParent

Returns the image at the top of the hierarchy, or NULL if there is no image on top.

BitsPerPixel

Gets the number of storage bits per pixel.

ColorSystem**ColPitch**

Gets or sets the color system used by this image, as defined by the [EColorSystem](#) enumeration.

Comment

Gets or sets the Comment attribute of the ROI/image. This is a convenience property that allows storing some information in the ROI/image. Please note that not all file formats preserve this information.

Date

Gets or sets the Date attribute of the ROI/image. This is a convenience property that allows storing some information in the ROI/image. Please note that not all file formats preserve this information.

FirstSubROI

See GetFirstSubROI in the derived classes

Height

Gets or sets the height of the ROI.

IsAnROI

-

IsVoid

Tests whether if the topmost parent image of this hierarchy has a zero size.

[NextSiblingROI](#)

This method returns the ROI that is the next sibling of this ROI.

[OrgX](#)

Gets or sets the x-coordinate of the upper left corner of the ROI, relative to its parent ROI/image.

[OrgY](#)

Gets or sets the y-coordinate of the upper left corner of the ROI, relative to its parent ROI/image.

[Parent](#)

Returns the hierarchical parent of this object.

[PlanesPerPixel](#)

Gets the number of color components in each pixel of the ROI/image.

[RowPitch](#)

Gets the pitch of a row (the number of bytes between two vertically adjacent pixels). This is also valid for BW1 images (one bit per pixel).

[Title](#)

Gets or sets the Title attribute of the ROI/image. This is a convenience property that allows storing some information in the ROI/image. Please note that not all file formats preserve this information.

[TotalHeight](#)

Gets the height, in pixels, of the ROI topmost parent.

TotalOrgX	Gets the x-coordinate of the upper left pixel of the ROI, relative to its topmost parent upper left pixel.
TotalOrgY	Gets the y-coordinate of the upper left pixel of the ROI, relative to its topmost parent upper left pixel.
TotalWidth	Gets the width, in pixels, of the ROI topmost parent.
Type	Gets the ROI/image pixel type, as defined by the EImageType enumeration.
Width	M Gets or sets the width of the ROI. E
THODS	
Attach	This method attaches the ROI to another ROI or image. Only ROIs can be attached. An image can never be attached to another image or ROI. Optionally, the ROI can be moved and resized after attachment by supplying additional parameters.
CopyTo	Copies all the data of the current EBaseROI object into another EBaseROI object and returns it.

[CropToImage](#)

	This method reduces the ROI size so that it is completely contained within its topmost parent image bounds (if such an image exists at the top of the hierarchy). This means that, after calling this method, either the ROI has a zero size, or all of its pixels map to valid pixels of the underlying image buffer. If the ROI is already contained within its parent image bounds, then this method does nothing.
Drag	Moves the specified handle to a new position and updates all placement parameters of the ROI.
Draw	Draws an ROI/image in a device context.
DrawFrame	Draws a rectangular frame around an image or ROI.
DrawFrameWithCurrentPen	Draws a rectangular frame around an image or ROI.
GetImagePtr	Returns a pointer to the pixel at given coordinates within the image/ROI.
GetSubBaseROIs	Returns all the children, and possibly recursively all their children, too.
HasSubROI	Tests whether this object has attached ROIs.

HitTest	Detects if the cursor is placed over one of the dragging handles.
Load	Restores an image stored in the given file.
Save	Saves the EBaseROI object to the given file.
SaveJpeg	Saves the EBaseROI object to the given file, in JPEG format.
SaveJpeg2K	Saves the EBaseROI object to the given file, in JPEG 2000 format.
Serialize	-
SetImagePtr	Sets the pointer to an externally allocated image buffer.
SetPlacement	Sets the placement of an ROI, relative to its parent ROI/image.
SetSize	Sets the width and height of an ROI/image.

EBaseROI.Attach

This method attaches the ROI to another ROI or image. Only ROIs can be attached. An image can never be attached to another image or ROI. Optionally, the ROI can be moved and resized after attachment by supplying additional parameters.

```
[VB6]

void Attach(
    EBaseROI parent
)

void Attach(
    EBaseROI parent,
    Long orgX,
    Long orgY,
    Long width,
    Long height
)
```

Parameters

parent

-

orgX

When specified, sets the new x-coordinate of the ROI top-left corner.

orgY

When specified, sets the new y-coordinate of the ROI top-left corner.

width

When specified, sets the new width of the ROI.

height

When specified, sets the new height of the ROI.

EBaseROI.Author

Gets or sets the Author attribute of the ROI/image. This is a convenience property that allows storing some information in the ROI/image. Please note that not all file formats preserve this information.

[VB6]

Author As String

read-write

EBaseROI.BaseTopParent

Returns the image at the top of the hierarchy, or NULL if there is no image on top.

[VB6]

BaseTopParent As EBaseROI

read-only

EBaseROI.BitsPerPixel

Gets the number of storage bits per pixel.

[VB6]

BitsPerPixel As Long

read-only

EBaseROI.ColorSystem

Gets or sets the color system used by this image, as defined by the [EColorSystem](#) enumeration.

[VB6]

ColorSystem As EColorSystem

read-write

Remarks

Upon object creation, a default color system is set, compatible with the ROI/image type ([EColorSystem::EColorSystem_GrayLevel](#) for gray-level types and [EColorSystem::EColorSystem_Rgb](#) for color types). The color system associated to an image is mainly relevant when working on color images. See [EasyColor \(FG\)](#) for more information.

EBaseROI.ColPitch

Gets the pitch of a column (number of bytes between two horizontally adjacent pixels). For BW1 (one bit per pixel) images, this value is undefined.

[VB6]

ColPitch As Long

read-only

EBaseROI.Comment

Gets or sets the Comment attribute of the ROI/image. This is a convenience property that allows storing some information in the ROI/image. Please note that not all file formats preserve this information.

[VB6]

Comment As String

read-write

EBaseROI.CopyTo

Copies all the data of the current EBaseROI object into another EBaseROI object and returns it.

[VB6]

```
void CopyTo(
    EBaseROI dest
)
```

Parameters

dest

Pointer to the EBaseROI object in which the current EBaseROI object data have to be copied.

Remarks

This method copies all the object date to the destination object. The attached ROIs are copied recursively and attached to the destination object. They will be deleted automatically when the the destination object is deleted. When the buffer of the source image has been provided by a call to **SetImagePtr**, the pointer will be copied into the destination image. Both images will thus refer the same external buffer.

EBaseROI.CropToImage

This method reduces the ROI size so that it is completely contained within its topmost parent image bounds (if such an image exists at the top of the hierarchy). This means that, after calling this method, either the ROI has a zero size, or all of its pixels map to valid pixels of the underlying image buffer. If the ROI is already contained within its parent image bounds, then this method does nothing.

[VB6]

```
void CropToImage(
)
```

EBaseROI.Date

Gets or sets the Date attribute of the ROI/image. This is a convenience property that allows storing some information in the ROI/image. Please note that not all file formats preserve this information.

[VB6]

Date As String

read-write

EBaseROI.Drag

Moves the specified handle to a new position and updates all placement parameters of the ROI.

[VB6]

```
void Drag(
    EDragHandle eHandle,
    Long n32X,
    Long n32Y,
    Single f32ZoomX,
    Single f32ZoomY,
    Single f32PanX,
    Single f32PanY
)
```

Parameters

eHandle

-

n32X

-

n32Y

-

f32ZoomX

-
f32ZoomY

-
f32PanX

-
f32PanY

Remarks

If zooming and/or panning were used when drawing the ROI, the same values must be used with [EBaseROI::HitTest](#) and [EBaseROI::Drag](#).

EBaseROI.Draw

Draws an ROI/image in a device context.

[VB6]

```
void Draw(
    EDrawAdapter graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void Draw(
    EDrawAdapter graphicContext,
    EC24Vector c24Vector,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void Draw(
    EDrawAdapter graphicContext,
    EBW8Vector bw8Vector,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

```

void Draw(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void Draw(
    Long graphicContext,
    EC24Vector c24Vector,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void Draw(
    Long graphicContext,
    EBW8Vector bw8Vector,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

```

Parameters

graphicContext

Handle to the device context of the destination window.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the image is displayed in **TRUE** scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a **0** value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Pan offset (in pixels) in the horizontal direction. By default, no panning is applied.

panY

Pan offset (in pixels) in the vertical direction. By default, no panning is applied.

c24Vector

When supplied, this parameter allows using a LUT that maps from BW8 to C24 when drawing (false colors).

bw8Vector

When supplied, this parameter allows using a LUT that maps from BW8 to BW8 when drawing.

Remarks

An ROI/image can be drawn (its pixels rendered) using a device context. The horizontal and vertical zooming factors can be different and must be contained in the **1/16..16** range. (MFC users can use the **CDC::GetSafeHdc()** method to obtain a suitable device context handle from a **CDC** instance.)

EBaseROI.DrawFrame

Draws a rectangular frame around an image or ROI.

[VB6]

```
void DrawFrame(
    EDrawAdapter graphicContext,
    EFramePosition framePosition,
    Boolean handles,
    Single zoomX,
    Single zoomY,
    Single x,
    Single y
)

void DrawFrame(
    EDrawAdapter graphicContext,
    Boolean handles,
    Single zoomX,
    Single zoomY,
    Single x,
    Single y
)

void DrawFrame(
    Long graphicContext,
    EFramePosition framePosition,
    Boolean bHandles,
    Single zoomX,
    Single zoomY,
    Single x,
    Single y
)
```

```
void DrawFrame(
    Long graphicContext,
    Boolean bHandles,
    Single zoomX,
    Single zoomY,
    Single x,
    Single y
)

void DrawFrame(
    Long graphicContext,
    ERGBColor color,
    Boolean bHandles,
    Single zoomX,
    Single zoomY,
    Single x,
    Single y
)
```

Parameters

graphicContext

Handle to the device context of the destination window.

framePosition

-

handles

TRUE if handles are to be drawn.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the image is displayed in **TRUE** scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a **0** value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

x

Translation factor for panning in the horizontal direction.

y

Translation factor for panning in the vertical direction.

bHandles

-

color

-

Remarks

A frame can be drawn (using a device context) around an image or ROI, possibly with 9 sizing handles. A suitable default pen is used (see [EBaseROI::DrawFrameWithCurrentPen](#) if you wish to use the pen currently selected into the device context). Zooming and panning are possible. Please note that panning is applied *before* zooming. (MFC users can use the [CDC::GetSafeHdc](#) () method to obtain a suitable device context handle from a [CDC](#) instance.)

EBaseROI.DrawFrameWithCurrentPen

Draws a rectangular frame around an image or ROI.

[VB6]

```
void DrawFrameWithCurrentPen(
    Long graphicContext,
    Boolean bHandles,
    Single zoomX,
    Single zoomY,
    Single x,
    Single y
)

void DrawFrameWithCurrentPen(
    Long graphicContext,
    EFramePosition framePosition,
    Boolean bHandles,
    Single zoomX,
    Single zoomY,
    Single x,
    Single y
)
```

Parameters

graphicContext

Handle to the device context of the destination window.

bHandles

-

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the image is displayed in **TRUE** scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a **0** value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

x

Translation factor for panning in the horizontal direction.

y

Translation factor for panning in the vertical direction.

framePosition

-

Remarks

A frame can be drawn (using a device context) around an image or ROI, possibly with 9 sizing handles. The current device context pen is used. Zooming and panning are possible. Please note that panning is applied *before* zooming. (MFC users can use the **CDC::GetSafeHdc()** method to obtain a suitable device context handle from a **CDC** instance.)

EBaseROI.FirstSubROI

See GetFirstSubROI in the derived classes

[VB6]

FirstSubROI As EBaseROI

read-only

EBaseROI.GetImagePtr

Returns a pointer to the pixel at given coordinates within the image/ROI.

[VB6]

```
Long GetImagePtr(
    Long x,
    Long y
)
```

```

Long GetImagePtr(
    Long x,
    Long y
)
Long GetImagePtr(
)
Long GetImagePtr(
)

```

Parameters

x

The pixel x-coordinate.

y

The pixel y-coordinate.

Remarks

This methods returns the memory address of the byte that contains the pixel (or address that contains the first byte of the pixel if it is bigger than one byte). If the pixel coordinates are not specified, the method returns the address of the top-left pixel of the ROI/image.

EBaseROI.GetSubBaseROIs

Returns all the children, and possibly recursively all their children, too.

```

[VB6]
() EBaseROI GetSubBaseROIs(
    Boolean recursive
)
() EBaseROI GetSubBaseROIs(
    Boolean bRecursive
)

```

Parameters

recursive

TRUE to retrieve all sub-ROIs recursively. FALSE otherwise.

bRecursive

EBaseROI.HasSubROI

Tests whether this object has attached ROIs.

[VB6]

```
Boolean HasSubROI(  
    EBaseROI subROI  
)
```

Parameters

subROI

EBaseROI.Height

Gets or sets the height of the ROI.

[VB6]

Height As Long

read-write

Remarks

The *placement of an ROI* is given by the coordinates its upper left corner (origin point), relative to its parent ROI/image and by its size (width and height).

EBaseROI.HitTest

Detects if the cursor is placed over one of the dragging handles.

[VB6]

```
EDragHandle HitTest(  
    Long x,  
    Long y,  
    Single zoomX,  
    Single zoomY,  
    Single panX,  
    Single panY  
)
```

Parameters

x

x-coordinate of the mouse cursor.

y

y-coordinate of the mouse cursor.

zoomX

Must be set to the same horizontal zooming factor as the one used when drawing. By default, true scale is used.

zoomY

Must be set to the same vertical zooming factor as the one used when drawing. By default, true scale is used.

panX

Must be set to the same horizontal pan offset factor as the one used when drawing. By default, true scale is used.

panY

Must be set to the same vertical pan offset as the one used when drawing. By default, true scale is used.

Remarks

Returns a handle identifier, as defined by [EDragHandle](#). If zooming and/or panning were used when drawing the ROI, the same values must be used with [EBaseROI::HitTest](#) and [EBaseROI::Drag](#).

EBaseROI.IsAnROI

[VB6]

IsAnROI As Boolean

read-only

EBaseROI.IsVoid

Tests whether if the topmost parent image of this hierarchy has a zero size.

[VB6]

IsVoid As Boolean

read-only

Remarks

For an image, this method returns **TRUE** if the image size is zero. For an ROI, this method returns **TRUE** if the topmost parent image size is zero or if there is no topmost image.

EBaseROI.Load

Restores an image stored in the given file.

[VB6]

```
void Load(  
    String path  
)  
  
void Load(  
    ESerializer serializer  
)
```

Parameters

path

Full path of the file.

serializer

The [ESerializer](#) file-like object that is read from.

Remarks

When loading, an image is resized if need be. On the opposite, an ROI cannot be resized, and the sizes *must* match. The image contents around the ROI remains unchanged. If a serializer is used, then the Euresys proprietary file format is expected. This format preserves attributes and sub-ROIs. See [Supported Image File Types](#) for details about supported files. See [Image File Access - Save, Load](#) - for details about file format and compatibility.

EBaseROI.NextSiblingROI

This method returns the ROI that is the next sibling of this ROI.

[VB6]

NextSiblingROI As EBaseROI

read-only

EBaseROI.OrgX

Gets or sets the x-coordinate of the upper left corner of the ROI, relative to its parent ROI/image.

[VB6]

OrgX As Long

read-write

Remarks

The *placement* of an ROI is given by the coordinates its upper left corner (origin point), relative to its parent ROI/image and by its size (width and height).

EBaseROI.OrgY

Gets or sets the y-coordinate of the upper left corner of the ROI, relative to its parent ROI/image.

[VB6]

OrgY As Long

read-write

Remarks

The *placement of an ROI* is given by the coordinates its upper left corner (origin point), relative to its parent ROI/image and by its size (width and height).

EBaseROI.Parent

Returns the hierarchical parent of this object.

[VB6]

Parent As EBaseROI

read-only

EBaseROI.PlanesPerPixel

Gets the number of color components in each pixel of the ROI/image.

[VB6]

PlanesPerPixel As Long

read-only

EBaseROI.RowPitch

Gets the pitch of a row (the number of bytes between two vertically adjacent pixels). This is also valid for BW1 images (one bit per pixel).

[VB6]

RowPitch As Long

read-only

EBaseROI.Save

Saves the EBaseROI object to the given file.

[VB6]

```
void Save(  
    String path,  
    EImageFileType type  
)  
  
void Save(  
    ESerializer serializer  
)
```

Parameters

path

The full path of the destination file.

type

File format, as defined by [EImageFileType](#). If not specified, the file format is determined from the file extension.

serializer

The [ESerializer](#) file-like object that is written to.

Remarks

By default (if no format is specified), the file format is determined from the file extension. If a serializer is used, then the Euresys proprietary file format is used. This format preserves attributes and sub-ROIs. See [Supported Image File Types](#) for details about supported files. See [Image File Access - Save, Load -](#) for details about file format and compatibility.

EBaseROI.SaveJpeg

Saves the EBaseROI object to the given file, in JPEG format.

[VB6]

```
void SaveJpeg(  
    String path,  
    Long quality  
)
```

Parameters

path

The full path of the destination file.

quality

JPEG quality, between 0 and 100 (100 is best quality). The default value is 75.

Remarks

See [Image File Access - Save, Load -](#) for details about file format and quality.

EBaseROI.SaveJpeg2K

Saves the EBaseROI object to the given file, in JPEG 2000 format.

[VB6]

```
void SaveJpeg2K(  
    String path,  
    Long quality  
)
```

Parameters

path

The full path of the destination file.

quality

JPEG 2000 quality, between 1 and 512. The default value is 16.

Remarks

See Image File Access - Save, Load - for details about file format and quality.

EBaseROI.Serialize

[VB6]

```
void Serialize(  
    ESerializer serializer  
)
```

Parameters

serializer

EBaseROI.SetImagePtr

Sets the pointer to an externally allocated image buffer.

[VB6]

```
void SetImagePtr(
    Long width,
    Long height,
    Long imagePointer,
    Long bitsPerRow
)
```

Parameters

width

The width of the supplied buffer, in pixels.

height

The height of the supplied buffer, in pixels.

imagePointer

The pointer (aligned on 4 bytes) to the buffer, which must be large enough to hold the image.

bitsPerRow

The total number of bits contained in a row, padding included. Using the value **0** (default) means that this size is computed from the buffer width and the pixel size plus a padding with the smallest possible value that leads to a multiple of 4 bytes (32 bits), which is the minimum padding accepted by [SetImagePtr](#).

Remarks

This call is only valid on an image. An ROI gets its buffer from its parent while an image normally allocates a pixel buffer automatically. The pointer to this buffer refers to the top left pixel of the image. The next pixels are stored contiguously, row by row, from top to bottom and from left to right. Padding at the end of a row may be used, but it must lead to rows that are multiple of 4 bytes. This method overrides the internally allocated image buffer of the [EBaseROI](#). As long as the image accesses this buffer, it must not be deleted.

EBaseROI.SetPlacement

Sets the placement of an ROI, relative to its parent ROI/image.

[VB6]

```
void SetPlacement(
    Long x,
    Long y,
    Long w,
    Long h
)
```

Parameters

x

-

y

-

w

-

h

-

Remarks

This method can only be called on ROIs. The *placement of an ROI* is given by the coordinates its upper left corner (origin point), relative to its parent ROI/image and by its size (width and height).

EBaseROI.GetSize

Sets the width and height of an ROI/image.

[VB6]

```
void SetSize(
    Long width,
    Long height
)

void SetSize(
    EBaseROI other
)
```

Parameters*width*

The new requested ROI/image width.

height

The new requested ROI/image height.

other

The other ROI/image whose dimensions have to be used for the current object.

Remarks

Open eVision will allocate a new image buffer (deallocate the old image buffer) if the supplied width and height are different from the existing ones. If an external buffer has been specified by means of **SetImagePtr**, it will be kept only if the size does not change. Creating a new Open eVision image buffer and setting its size creates a 4-byte aligned buffer, by default. The *size of an image* is specified as a number of columns (*width*) and rows (*height*). The maximum image dimensions are 32767 by 32767. Furthermore, it must fit into the available memory, that depends upon the physical memory, the operating system and the memory already allocated by the process in other modules or libraries. The *placement of an ROI* is given by the *x* and *y* coordinates of its upper left pixel relative to its parent image, and also by its width and its height.

EBaseROI.Title

Gets or sets the Title attribute of the ROI/image. This is a convenience property that allows storing some information in the ROI/image. Please note that not all file formats preserve this information.

[VB6]

Title As String

read-write

EBaseROI.TotalHeight

Gets the height, in pixels, of the ROI topmost parent.

[VB6]

TotalHeight As Long

read-only

Remarks

The *total size* of an ROI is the size of its *topmost* parent.

EBaseROI.TotalOrgX

Gets the x-coordinate of the upper left pixel of the ROI, relative to its topmost parent upper left pixel.

[VB6]

TotalOrgX As Long

read-only

Remarks

The *total origin coordinates* of an ROI indicate the position of its upper left pixel relative to the *topmost* parent image. The total origin coordinates (top-left pixel) of a topmost parent are always **(0,0)**.

EBaseROI.TotalOrgY

Gets the y-coordinate of the upper left pixel of the ROI, relative to its topmost parent upper left pixel.

[VB6]

TotalOrgY As Long

read-only

Remarks

The *total origin coordinates* of an ROI indicate the position of its upper left pixel relative to the *topmost* parent. The total origin coordinates (top-left pixel) of a topmost parent are always **(0,0)**.

EBaseROI.TotalWidth

Gets the width, in pixels, of the ROI topmost parent.

[VB6]

TotalWidth As Long

read-only

Remarks

The *total size* of an ROI is the size of its *topmost* parent.

EBaseROI.Type

Gets the ROI/image pixel type, as defined by the [EImageType](#) enumeration.

[VB6]

Type As EImageType

read-only

EBaseROI.Width

Gets or sets the width of the ROI.

[VB6]

Width As Long

read-write

Remarks

The placement of an ROI is given by the coordinates its upper left corner (origin point), relative to its parent ROI/image and by its size (width and height).

EBinaryImageSegmenter Class

Segments a binary image.

Remarks

This segmenter is applicable to [EROIBW1](#) grayscale images. It produces coded images with two layers: The Black layer (usually, with index 0) contains the unmasked pixels having a binary value equal to zero; and the White layer (usually, with index 1) contains the remaining unmasked pixels, i.e. unmasked pixels having a binary value equal to one.

Base Class: [ETwoLayersImageSegmenter](#)

EBW16PathVector Class

Vector objects are used to store 1-dimensional data.

Remarks

Using vectors is very similar to using 1-dimensional arrays, except that the size can vary at runtime. Memory allocation is handled internally. * To create a vector, use its constructor. * To fill a vector with values, first empty it, using the [EVector::Empty](#) member, and then add elements one at time at the tail by calling the [EBW16PathVector::AddElement](#) member. * To access a vector element, either for reading or writing, use the [EBW16PathVector.operator\[\]@](#). * To inquire for the current number of elements, use member [EVector::NumElements](#).

Base Class: [EVector](#)

PROPERTIES**Closed**

-

RawDataPtr**M**

Pointer to the vector data.

E**THODS****AddElement**

Appends (adds at the tail) an element to the vector.

Draw

Draws a plot of the vector element values.

DrawWithCurrentPen

Draws a plot of the vector element values.

EBW16PathVector

Constructs a vector.

GetElement

Returns the vector element at the given index.

operator[]

Gives access to the vector element at the given index.

operator=

Copies all the data from another EBW16PathVector object into the current EBW16PathVector object

SetElement

E Modifies the vector element at the given index by the given value.
B

W16PathVector.AddElement

Appends (adds at the tail) an element to the vector.

[VB6]

```
void AddElement(  
    EBW16Path element  
)
```

Parameters

element

The element to be added.

EBW16PathVector.Closed

-

[VB6]

Closed As Boolean
read-write

EBW16PathVector.Draw

Draws a plot of the vector element values.

[VB6]

```

void Draw(
    EDrawAdapter graphicContext,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

void Draw(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

void Draw(
    Long graphicContext,
    ERGBColor color,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

```

Parameters

graphicContext

Handle of the device context on which to draw.

zoomX

Zooming factor along the X axis (**1.0f** means no zoom).

zoomY

Zooming factor along the Y axis (**1.0f** means no zoom).

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

color

The color in which to draw the overlay.

Remarks

The vector draws line segment between the element coordinates. The drawing appears on the image itself.

EBW16PathVector.DrawWithCurrentPen

Draws a plot of the vector element values.

[VB6]

```
void DrawWithCurrentPen(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

zoomX

Zooming factor along the X axis (**1.0f** means no zoom).

zoomY

Zooming factor along the Y axis (**1.0f** means no zoom).

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

Remarks

The vector draws line segment between the element coordinates. The drawing appears on the image itself.

EBW16PathVector.EBW16PathVector

Constructs a vector.

```
[VB6]  
void EBW16PathVector()  
)  
  
void EBW16PathVector(  
Long maxNumberOfElements  
)  
  
void EBW16PathVector(  
EBW16PathVector other  
)
```

Parameters

maxNumberOfElements

Optionally, memory can be pre-allocated to accommodate a given number of elements.

other

EBW16PathVector object to be copied

EBW16PathVector.GetElement

Returns the vector element at the given index.

```
[VB6]  
EBW16Path GetElement(  
Long index  
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded) of the element to be accessed.

Remarks

If the given index is outside the bounds of the vector, the error code [EError_Parameter1OutOfRange](#) is set.

EBW16PathVector.operator[]

Gives access to the vector element at the given index.

[VB6]

```
EBW16Path operator[] (
    Long index
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded) of the element to be accessed.

EBW16PathVector.operator=

Copies all the data from another EBW16PathVector object into the current EBW16PathVector object

[VB6]

```
EBW16PathVector operator=(
    EBW16PathVector other
)
```

Parameters

other

EBW16PathVector object to be copied

EBW16PathVector.RawDataPtr

Pointer to the vector data.

[VB6]

RawDataPtr As Long

read-only

EBW16PathVector.SetElement

Modifies the vector element at the given index by the given value.

[VB6]

```
void SetElement(  
    Long index,  
    EBW16Path value  
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded), of the element to be modified.

value

The new value for the element.

Remarks

If the given index is outside the bounds of the vector, the error code [EError_Parameter1OutOfRange](#) is set.

EBW16Vector Class

Vector objects are used to store 1-dimensional data.

Remarks

Using vectors is very similar to using 1-dimensional arrays, except that the size can vary at runtime. Memory allocation is handled internally. * To create a vector, use its constructor. * To fill a vector with values, first empty it, using the [EVector::Empty](#) member, and then add elements one at time at the tail by calling the [EBW16Vector::AddElement](#) member. * To access a vector element, either for reading or writing, use the [EBW16Vector.operator\[\]@](#). * To inquire for the current number of elements, use member [EVector::NumElements](#).

Base Class: [EVector](#)

PROPERTIES

[RawDataPtr](#)

M

Pointer to the vector data.

THODS

[AddElement](#)

Appends (adds at the tail) an element to the vector.

[Draw](#)

Draws a plot of the vector element values.

[DrawWithCurrentPen](#)

Draws a plot of the vector element values.

[EBW16Vector](#)

Constructs a vector.

[GetElement](#)

Returns the vector element at the given index.

<code>operator[]</code>	Gives access to the vector element at the given index.
<code>operator=</code>	Copies all the data from another EBW16Vector object into the current EBW16Vector object
<code>SetElement</code>	Modifies the vector element at the given index by the given value.
<code>WeightedMoment</code>	Returns the first order geometric moment (weighted gravity center).
B	

W16Vector.AddElement

Appends (adds at the tail) an element to the vector.

```
[VB6]
void AddElement(
    EBW16 element
)
```

Parameters

`element`

The element to be added.

EBW16Vector.Draw

Draws a plot of the vector element values.

[VB6]

```
void Draw(
    EDrawAdapter graphicContext,
    Single width,
    Single height,
    Single originX,
    Single originY
)

void Draw(
    Long graphicContext,
    Single width,
    Single height,
    Single originX,
    Single originY
)

void Draw(
    Long graphicContext,
    ERGBColor color,
    Single width,
    Single height,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

width

Outermost horizontal size, in pixels.

height

Outermost vertical size, in pixels.

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

color

The color in which to draw the overlay.

Remarks

A vector is able to draw itself in a window. The vector plots the element values as a function of the element indices. The drawing appears on a neutral background.

EBW16Vector.DrawWithCurrentPen

Draws a plot of the vector element values.

[VB6]

```
void DrawWithCurrentPen(
    Long graphicContext,
    Single width,
    Single height,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

width

Outermost horizontal size, in pixels.

height

Outermost vertical size, in pixels.

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

Remarks

A vector is able to draw itself in a window. The vector plots the element values as a function of the element indices. The drawing appears on a neutral background.

EBW16Vector.EBW16Vector

Constructs a vector.

```
[VB6]
void EBW16Vector(
)
void EBW16Vector(
    Long maxNumberOfElements
)
void EBW16Vector(
    EBW16Vector other
)
```

Parameters

maxNumberOfElements

Optionally, memory can be pre-allocated to accommodate a given number of elements.

other

EBW16Vector object to be copied

EBW16Vector.GetElement

Returns the vector element at the given index.

```
[VB6]
EBW16 GetElement(
    Long index
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded) of the element to be accessed.

Remarks

If the given index is outside the bounds of the vector, the error code [EError_Parameter1OutOfRange](#) is set.

EBW16Vector.operator[]

Gives access to the vector element at the given index.

[VB6]

```
EBW16 operator[](  
    Long index  
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded) of the element to be accessed.

EBW16Vector.operator=

Copies all the data from another EBW16Vector object into the current EBW16Vector object

[VB6]

```
EBW16Vector operator=(  
    EBW16Vector other  
)
```

Parameters

other

EBW16Vector object to be copied

EBW16Vector.RawDataPtr

Pointer to the vector data.

[VB6]

RawDataPtr As Long

read-only

EBW16Vector.SetElement

Modifies the vector element at the given index by the given value.

[VB6]

```
void SetElement(  
    Long index,  
    EBW16 value  
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded), of the element to be modified.

value

The new value for the element.

Remarks

If the given index is outside the bounds of the vector, the error code [EError_Parameter1OutOfRange](#) is set.

EBW16Vector.WeightedMoment

Returns the first order geometric moment (weighted gravity center).

[VB6]

```
Single WeightedMoment(
    Long from,
    Long to
)
```

Parameters

from

First element of the vector portion for which the weighted moment will be calculated. By default, this is the first element of the vector.

to

Last element of the vector portion for which the weighted moment will be calculated. By default, this is the last element of the vector.

EBW32Vector Class

Vector objects are used to store 1-dimensional data.

Remarks

Using vectors is very similar to using 1-dimensional arrays, except that the size can vary at runtime. Memory allocation is handled internally. * To create a vector, use its constructor. * To fill a vector with values, first empty it, using the [EVector::Empty](#) member, and then add elements one at time at the tail by calling the [EBW32Vector::AddElement](#) member. * To access a vector element, either for reading or writing, use the [EBW32Vector.operator\[\]@](#). * To inquire for the current number of elements, use member [EVector::NumElements](#).

Base Class: [EVector](#)

PROPERTIES

RawDataPtr

Pointer to the vector data.

METHODS

AddElement	Appends (adds at the tail) an element to the vector.
Draw	Draws a plot of the vector element values.
DrawWithCurrentPen	Draws a plot of the vector element values.
EBW32Vector	Constructs a vector.
GetElement	Returns the vector element at the given index.
operator[]	Gives access to the vector element at the given index.
operator=	Copies all the data from another EBW32Vector object into the current EBW32Vector object
SetElement	Modifies the vector element at the given index by the given value.
WeightedMoment	Returns the first order geometric moment (weighted gravity center).

EBW32Vector.AddElement

Appends (adds at the tail) an element to the vector.

```
[VB6]  
void AddElement(  
    EBW32 element  
)
```

Parameters

element
The element to be added.

EBW32Vector.Draw

Draws a plot of the vector element values.

```
[VB6]  
void Draw(  
    EDrawAdapter graphicContext,  
    Single width,  
    Single height,  
    Single originX,  
    Single originY  
)  
  
void Draw(  
    Long graphicContext,  
    Single width,  
    Single height,  
    Single originX,  
    Single originY  
)
```

```
void Draw(
    Long graphicContext,
    ERGBColor color,
    Single width,
    Single height,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

width

Outermost horizontal size, in pixels.

height

Outermost vertical size, in pixels.

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

color

The color in which to draw the overlay.

Remarks

A vector is able to draw itself in a window. The vector plots the element values as a function of the element indices. The drawing appears on a neutral background.

EBW32Vector.DrawWithCurrentPen

Draws a plot of the vector element values.

[VB6]

```
void DrawWithCurrentPen(
    Long graphicContext,
    Single width,
    Single height,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

width

Outermost horizontal size, in pixels.

height

Outermost vertical size, in pixels.

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

Remarks

A vector is able to draw itself in a window. The vector plots the element values as a function of the element indices. The drawing appears on a neutral background.

EBW32Vector.EBW32Vector

Constructs a vector.

[VB6]

```
void EBW32Vector(
)
```

```

void EBW32Vector(
    Long maxNumberOfElements
)
void EBW32Vector(
    EBW32Vector other
)

```

Parameters*maxNumberOfElements*

Optionally, memory can be pre-allocated to accommodate a given number of elements.

other

EBW32Vector object to be copied

EBW32Vector.GetElement

Returns the vector element at the given index.

[VB6]

```

EBW32 GetElement(
    Long index
)

```

Parameters*index*

Index, between **0** and [EVector::NumElements](#) (excluded) of the element to be accessed.

Remarks

If the given index is outside the bounds of the vector, the error code [EError_Parameter1OutOfRange](#) is set.

EBW32Vector.operator[]

Gives access to the vector element at the given index.

[VB6]

```
EBW32 operator[](  
    Long index  
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded) of the element to be accessed.

EBW32Vector.operator=

Copies all the data from another EBW32Vector object into the current EBW32Vector object

[VB6]

```
EBW32Vector operator=(  
    EBW32Vector other  
)
```

Parameters

other

EBW32Vector object to be copied

EBW32Vector.RawDataPtr

Pointer to the vector data.

[VB6]

```
RawDataPtr As Long  
read-only
```

EBW32Vector.SetElement

Modifies the vector element at the given index by the given value.

```
[VB6]
void SetElement(
    Long index,
    EBW32 value
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded), of the element to be modified.

value

The new value for the element.

Remarks

If the given index is outside the bounds of the vector, the error code [EError_Parameter1OutOfRange](#) is set.

EBW32Vector.WeightedMoment

Returns the first order geometric moment (weighted gravity center).

```
[VB6]
Single WeightedMoment(
    Long from,
    Long to
)
```

Parameters

from

First element of the vector portion for which the weighted moment will be calculated. By default, this is the first element of the vector.

to

Last element of the vector portion for which the weighted moment will be calculated. By default, this is the last element of the vector.

EBW8PathVector Class

Vector objects are used to store 1-dimensional data.

Remarks

Using vectors is very similar to using 1-dimensional arrays, except that the size can vary at runtime. Memory allocation is handled internally. * To create a vector, use its constructor. * To fill a vector with values, first empty it, using the [EVector::Empty](#) member, and then add elements one at time at the tail by calling the [EBW8PathVector::AddElement](#) member. * To access a vector element, either for reading or writing, use the [EBW8PathVector.operator\[\]@](#). * To inquire for the current number of elements, use member [EVector::NumElements](#).

Base Class: [EVector](#)

PROPERTIES

[Closed](#)

Flag indicating whether the shape built with [EasyImage::Contour](#) must be closed or not.

[RawDataPtr](#)

M Pointer to the vector data.

THODS

[AddElement](#)

Appends (adds at the tail) an element to the vector.

[Draw](#)

Draws a plot of the vector element values.

DrawWithCurrentPen	Draws a plot of the vector element values.
EBW8PathVector	Constructs a vector.
GetElement	Returns the vector element at the given index.
operator[]	Gives access to the vector element at the given index.
operator=	Copies all the data from another EBW8PathVector object into the current EBW8PathVector object
SetElement	Modifies the vector element at the given index by the given value.

B

W8PathVector.AddElement

Appends (adds at the tail) an element to the vector.

```
[VB6]
void AddElement(
    EBW8Path element
)
```

Parameters

element

The element to be added.

EBW8PathVector.Closed

Flag indicating whether the shape built with [EasyImage::Contour](#) must be closed or not.

[VB6]

Closed As Boolean

read-write

EBW8PathVector.Draw

Draws a plot of the vector element values.

[VB6]

```
void Draw(
    EDrawAdapter graphicContext,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

void Draw(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

void Draw(
    Long graphicContext,
    ERGBColor color,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

Parameters*graphicContext*

Handle of the device context on which to draw.

*zoomX*Zooming factor along the X axis (**1.0f** means no zoom).*zoomY*Zooming factor along the Y axis (**1.0f** means no zoom).*originX*

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

color

The color in which to draw the overlay.

Remarks

The vector draws line segment between the element coordinates. The drawing appears on the image itself.

EBW8PathVector.DrawWithCurrentPen

Draws a plot of the vector element values.

[VB6]

```
void DrawWithCurrentPen(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

Parameters*graphicContext*

Handle of the device context on which to draw.

*zoomX*Zooming factor along the X axis (**1.0f** means no zoom).

zoomY

Zooming factor along the Y axis (**1.0f** means no zoom).

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

Remarks

The vector draws line segment between the element coordinates. The drawing appears on the image itself.

EBW8PathVector.EBW8PathVector

Constructs a vector.

```
[VB6]
void EBW8PathVector(
)
void EBW8PathVector(
    Long maxNumberOfElements
)
void EBW8PathVector(
    EBW8PathVector other
)
```

Parameters

maxNumberOfElements

Optionally, memory can be pre-allocated to accommodate a given number of elements.

other

EBW8PathVector object to be copied

EBW8PathVector.GetElement

Returns the vector element at the given index.

[VB6]

```
EBW8Path GetElement(  
    Long index  
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded) of the element to be accessed.

Remarks

If the given index is outside the bounds of the vector, the error code [EError_Parameter1OutOfRange](#) is set.

EBW8PathVector.operator[]

Gives access to the vector element at the given index.

[VB6]

```
EBW8Path operator[](  
    Long index  
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded) of the element to be accessed.

EBW8PathVector.operator=

Copies all the data from another EBW8PathVector object into the current EBW8PathVector object

[VB6]

```
EBW8PathVector operator=(  
    EBW8PathVector other  
)
```

Parameters

other

EBW8PathVector object to be copied

EBW8PathVector.RawDataPtr

Pointer to the vector data.

[VB6]

```
RawDataPtr As Long  
read-only
```

EBW8PathVector.SetElement

Modifies the vector element at the given index by the given value.

[VB6]

```
void SetElement(  
    Long index,  
    EBW8Path value  
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded), of the element to be modified.

value

The new value for the element.

Remarks

If the given index is outside the bounds of the vector, the error code [EError_Parameter1OutOfRange](#) is set.

EBW8Vector Class

Vector objects are used to store 1-dimensional data.

Remarks

Using vectors is very similar to using 1-dimensional arrays, except that the size can vary at runtime. Memory allocation is handled internally. * To create a vector, use its constructor. * To fill a vector with values, first empty it, using the [EVector::Empty](#) member, and then add elements one at time at the tail by calling the [EBW8Vector::AddElement](#) member. * To access a vector element, either for reading or writing, use the [EBW8Vector.operator\[\]@](#). * To inquire for the current number of elements, use member [EVector::NumElements](#).

Base Class: [EVector](#)

PROPERTIES

[RawDataPtr](#)

M

Pointer to the vector data.

THODS

[AddElement](#)

Appends (adds at the tail) an element to the vector.

[Draw](#)

Draws a plot of the vector element values.

[DrawWithCurrentPen](#)

Draws a plot of the vector element values.

EBW8Vector	Constructs a vector.
GetElement	Returns the vector element at the given index.
operator[]	Gives access to the vector element at the given index.
operator=	Copies all the data from another EBW8Vector object into the current EBW8Vector object
SetElement	Modifies the vector element at the given index by the given value.
WeightedMoment	E Returns the first order geometric moment (weighted gravity center). B

W8Vector.AddElement

Appends (adds at the tail) an element to the vector.

```
[VB6]
void AddElement(
    EBW8 element
)
```

Parameters

element

The element to be added.

EBW8Vector.Draw

Draws a plot of the vector element values.

[VB6]

```
void Draw(
    Long graphicContext,
    Single width,
    Single height,
    Single originX,
    Single originY
)

void Draw(
    EDrawAdapter graphicContext,
    Single width,
    Single height,
    Single originX,
    Single originY
)

void Draw(
    Long graphicContext,
    ERGBColor color,
    Single width,
    Single height,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

width

Outermost horizontal size, in pixels.

height

Outermost vertical size, in pixels.

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

color

The color in which to draw the overlay.

Remarks

A vector is able to draw itself in a window. The vector plots the element values as a function of the element indices. The drawing appears on a neutral background.

EBW8Vector.DrawWithCurrentPen

Draws a plot of the vector element values.

[VB6]

```
void DrawWithCurrentPen(
    Long graphicContext,
    Single width,
    Single height,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

width

Outermost horizontal size, in pixels.

height

Outermost vertical size, in pixels.

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

Remarks

A vector is able to draw itself in a window. The vector plots the element values as a function of the element indices. The drawing appears on a neutral background.

EBW8Vector.EBW8Vector

Constructs a vector.

```
[VB6]
void EBW8Vector(
)
void EBW8Vector(
    Long maxNumberOfElements
)
void EBW8Vector(
    EBW8Vector other
)
```

Parameters

maxNumberOfElements

Optionally, memory can be pre-allocated to accommodate a given number of elements.

other

EBW8Vector object to be copied

EBW8Vector.GetElement

Returns the vector element at the given index.

```
[VB6]
EBW8 GetElement(
    Long index
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded) of the element to be accessed.

Remarks

If the given index is outside the bounds of the vector, the error code [EError_Parameter1OutOfRange](#) is set.

EBW8Vector.operator[]

Gives access to the vector element at the given index.

[VB6]

```
EBW8 operator[] (
    Long index
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded) of the element to be accessed.

EBW8Vector.operator=

Copies all the data from another EBW8Vector object into the current EBW8Vector object

[VB6]

```
EBW8Vector operator=
    EBW8Vector other
)
```

Parameters

other

EBW8Vector object to be copied

EBW8Vector.RawDataPtr

Pointer to the vector data.

[VB6]

RawDataPtr As Long

read-only

EBW8Vector.SetElement

Modifies the vector element at the given index by the given value.

[VB6]

```
void SetElement(
    Long index,
    EBW8 value
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded), of the element to be modified.

value

The new value for the element.

Remarks

If the given index is outside the bounds of the vector, the error code [EError_Parameter1OutOfRange](#) is set.

EBW8Vector.WeightedMoment

Returns the first order geometric moment (weighted gravity center).

[VB6]

```
Single WeightedMoment(
    Long from,
    Long to
)
```

Parameters

from

First element of the vector portion for which the weighted moment will be calculated. By default, this is the first element of the vector.

to

Last element of the vector portion for which the weighted moment will be calculated. By default, this is the last element of the vector.

EBWHistogramVector Class

Vector objects are used to store 1-dimensional data.

Remarks

Using vectors is very similar to using 1-dimensional arrays, except that the size can vary at runtime. Memory allocation is handled internally. * To create a vector, use its constructor. * To fill a vector with values, first empty it, using the [EVector::Empty](#) member, and then add elements one at time at the tail by calling the [EBWHistogramVector::AddElement](#) member. * To access a vector element, either for reading or writing, use the [EBWHistogramVector.operator\[\]@](#). * To inquire for the current number of elements, use member [EVector::NumElements](#).

Base Class: [EVector](#)

PROPERTIES

RawDataPtr

Pointer to the vector data.

METHODS**AddElement**

Appends (adds at the tail) an element to the vector.

Draw

Draws a plot of the vector element values.

DrawWithCurrentPen

Draws a plot of the vector element values.

EBWHistogramVector

Constructs a vector.

GetElement

Returns the vector element at the given index.

operator[]

Gives access to the vector element at the given index.

operator=

Copies all the data from another EBWHistogramVector object into the current EBWHistogramVector object

SetElement

Modifies the vector element at the given index by the given value.

B

WHistogramVector.AddElement

Appends (adds at the tail) an element to the vector.

[VB6]

```
void AddElement(  
    Long element  
)
```

Parameters

element

The element to be added.

EBWHistogramVector.Draw

Draws a plot of the vector element values.

[VB6]

```
void Draw(  
    EDrawAdapter graphicContext,  
    Single width,  
    Single height,  
    Single originX,  
    Single originY  
)  
  
void Draw(  
    Long graphicContext,  
    Single width,  
    Single height,  
    Single originX,  
    Single originY  
)  
  
void Draw(  
    Long graphicContext,  
    ERGBColor color,  
    Single width,  
    Single height,  
    Single originX,  
    Single originY  
)
```

Parameters*graphicContext*

Handle of the device context on which to draw.

width

Outermost horizontal size, in pixels.

height

Outermost vertical size, in pixels.

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

color

The color in which to draw the overlay.

Remarks

A vector is able to draw itself in a window. The vector plots the element values as a function of the element indices. The drawing appears on a neutral background.

EBWHistogramVector.DrawWithCurrentPen

Draws a plot of the vector element values.

[VB6]

```
void DrawWithCurrentPen(
    Long graphicContext,
    Single width,
    Single height,
    Single originX,
    Single originY
)
```

Parameters*graphicContext*

Handle of the device context on which to draw.

width

Outermost horizontal size, in pixels.

height

Outermost vertical size, in pixels.

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

Remarks

A vector is able to draw itself in a window. The vector plots the element values as a function of the element indices. The drawing appears on a neutral background.

EBWHistogramVector.EBWHistogramVector

Constructs a vector.

```
[VB6]
void EBWHistogramVector()
)

void EBWHistogramVector(
    EBWHistogramVector other
)

void EBWHistogramVector(
    Long maxNumberOfElements
)
```

Parameters

other

EBWHistogramVector object to be copied

maxNumberOfElements

Optionally, memory can be pre-allocated to accommodate a given number of elements.

EBWHistogramVector.GetElement

Returns the vector element at the given index.

[VB6]

```
Long GetElement(
    Long index
)
```

Parameters*index*

Index, between **0** and [EVector::NumElements](#) (excluded) of the element to be accessed.

Remarks

If the given index is outside the bounds of the vector, the error code [EError_Parameter1OutOfRange](#) is set.

EBWHistogramVector.operator[]

Gives access to the vector element at the given index.

[VB6]

```
Long operator[] (
    Long index
)
```

Parameters*index*

Index, between **0** and [EVector::NumElements](#) (excluded) of the element to be accessed.

EBWHistogramVector.operator=

Copies all the data from another EBWHistogramVector object into the current EBWHistogramVector object

[VB6]

```
EBWHistogramVector operator=
    EBWHistogramVector other
)
```

Parameters*other*

EBWHistogramVector object to be copied

EBWHistogramVector.RawDataPtr

Pointer to the vector data.

[VB6]

RawDataPtr As Long

read-only

EBWHistogramVector.SetElement

Modifies the vector element at the given index by the given value.

[VB6]

```
void SetElement(
    Long index,
    Long value
)
```

Parameters*index*Index, between **0** and [EVector::NumElements](#) (excluded), of the element to be modified.

value

The new value for the element.

Remarks

If the given index is outside the bounds of the vector, the error code [EError_Parameter1OutOfRange](#) is set.

EC24PathVector Class

Vector objects are used to store 1-dimensional data.

Remarks

Using vectors is very similar to using 1-dimensional arrays, except that the size can vary at runtime. Memory allocation is handled internally. * To create a vector, use its constructor. * To fill a vector with values, first empty it, using the [EVector::Empty](#) member, and then add elements one at time at the tail by calling the [EC24PathVector::AddElement](#) member. * To access a vector element, either for reading or writing, use the [EC24PathVector.operator\[\]@](#). * To inquire for the current number of elements, use member [EVector::NumElements](#).

Base Class: [EVector](#)

PROPERTIES

Closed

-

RawDataPtr

M

Pointer to the vector data.

E

THODS

AddElement

Appends (adds at the tail) an element to the vector.

Draw

Draws a plot of the vector element values.

DrawWithCurrentPen	Draws a plot of the vector element values.
EC24PathVector	Constructs a vector.
GetElement	Returns the vector element at the given index.
operator[]	Gives access to the vector element at the given index.
operator=	Copies all the data from another EC24PathVector object into the current EC24PathVector object
SetElement	Modifies the vector element at the given index by the given value.

24PathVector.AddElement

Appends (adds at the tail) an element to the vector.

```
[VB6]
void AddElement(
    EC24Path element
)
```

Parameters

element

The element to be added.

EC24PathVector.Closed

-

[VB6]

Closed As Boolean

read-write

EC24PathVector.Draw

Draws a plot of the vector element values.

[VB6]

```
void Draw(
    EDrawAdapter graphicContext,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

void Draw(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

void Draw(
    Long graphicContext,
    ERGBColor color,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

Parameters*graphicContext*

Handle of the device context on which to draw.

zoomX

Zooming factor along the X axis (**1.0f** means no zoom).

zoomY

Zooming factor along the Y axis (**1.0f** means no zoom).

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

color

The color in which to draw the overlay.

Remarks

The vector draws line segment between the element coordinates. The drawing appears on the image itself.

EC24PathVector.DrawWithCurrentPen

Draws a plot of the vector element values.

[VB6]

```
void DrawWithCurrentPen(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

Parameters*graphicContext*

Handle of the device context on which to draw.

zoomX

Zooming factor along the X axis (**1.0f** means no zoom).

zoomY

Zooming factor along the Y axis (**1.0f** means no zoom).

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

Remarks

The vector draws line segment between the element coordinates. The drawing appears on the image itself.

EC24PathVector.EC24PathVector

Constructs a vector.

```
[VB6]
void EC24PathVector(
)
void EC24PathVector(
    EC24PathVector other
)
void EC24PathVector(
    Long maxNumberOfElements
)
```

Parameters

other

EC24PathVector object to be copied

maxNumberOfElements

Optionally, memory can be pre-allocated to accommodate a given number of elements.

EC24PathVector.GetElement

Returns the vector element at the given index.

[VB6]

```
EC24Path GetElement(  
    Long index  
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded) of the element to be accessed.

Remarks

If the given index is outside the bounds of the vector, the error code [EError_Parameter1OutOfRange](#) is set.

EC24PathVector.operator[]

Gives access to the vector element at the given index.

[VB6]

```
EC24Path operator[](  
    Long index  
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded) of the element to be accessed.

EC24PathVector.operator=

Copies all the data from another EC24PathVector object into the current EC24PathVector object

[VB6]

```
EC24PathVector operator=(  
    EC24PathVector other  
)
```

Parameters

other

EC24PathVector object to be copied

EC24PathVector.RawDataPtr

Pointer to the vector data.

[VB6]

```
RawDataPtr As Long  
read-only
```

EC24PathVector.SetElement

Modifies the vector element at the given index by the given value.

[VB6]

```
void SetElement(  
    Long index,  
    EC24Path value  
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded), of the element to be modified.

value

The new value for the element.

Remarks

If the given index is outside the bounds of the vector, the error code [EError_Parameter1OutOfRange](#) is set.

EC24Vector Class

Vector objects are used to store 1-dimensional data.

Remarks

Using vectors is very similar to using 1-dimensional arrays, except that the size can vary at runtime. Memory allocation is handled internally. * To create a vector, use its constructor. * To fill a vector with values, first empty it, using the [EVector::Empty](#) member, and then add elements one at time at the tail by calling the [EC24Vector::AddElement](#) member. * To access a vector element, either for reading or writing, use the [EC24Vector.operator\[\]@](#). * To inquire for the current number of elements, use member [EVector::NumElements](#).

Base Class: [EVector](#)

PROPERTIES

[RawDataPtr](#)



Pointer to the vector data.

THODS

[AddElement](#)

Appends (adds at the tail) an element to the vector.

[Draw](#)

Draws a plot of the vector element values.

[EC24Vector](#)

Constructs a vector.

GetElement

Returns the vector element at the given index.

operator[]

Gives access to the vector element at the given index.

operator=

Copies all the data from another EC24Vector object into the current EC24Vector object

SetElement

E Modifies the vector element at the given index by the given value.

C

24Vector.AddElement

Appends (adds at the tail) an element to the vector.

[VB6]

```
void AddElement(  
    EC24 element  
)
```

Parameters

element

The element to be added.

EC24Vector.Draw

Draws a plot of the vector element values.

[VB6]

```
void Draw(
    Long graphicContext,
    Single width,
    Single height
)

void Draw(
    Long graphicContext,
    Single width,
    Single height,
    Single originX,
    Single originY
)

void Draw(
    Long graphicContext,
    Single width,
    Single height,
    Single originX,
    Single originY,
    ERGBColor color0,
    ERGBColor color1,
    ERGBColor color2
)

void Draw(
    Long graphicContext,
    Single width,
    Single height,
    ERGBColor color0,
    ERGBColor color1,
    ERGBColor color2
)
```

```
void Draw(
    EDrawAdapter graphicContext,
    Single width,
    Single height
)

void Draw(
    EDrawAdapter graphicContext,
    Single width,
    Single height,
    Single originX,
    Single originY
)

void Draw(
    EDrawAdapter graphicContext,
    Single width,
    Single height,
    Single originX,
    Single originY,
    ERGBColor color0,
    ERGBColor color1,
    ERGBColor color2
)

void Draw(
    EDrawAdapter graphicContext,
    Single width,
    Single height,
    ERGBColor color0,
    ERGBColor color1,
    ERGBColor color2
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

width

Outermost horizontal size, in pixels.

height

Outermost vertical size, in pixels.

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

color0

The color to be used when drawing the curve of the first color component of the vector, as an RGB color. By default, the current pen is used to draw the curve.

color1

The color to be used when drawing the curve of the second color component of the vector, as an RGB color. By default, the current pen is used to draw the curve.

color2

The color to be used when drawing the curve of the third color component of the vector, as an RGB color. By default, the current pen is used to draw the curve.

Remarks

A vector is able to draw itself in a window. The vector plots the element values as a function of the element indices. The drawing appears on a neutral background. In the special case of the EC24Vector, three curves are drawn instead of one, each corresponding to a color component. Three pen objects must be provided to draw the curves with appropriate attributes.

EC24Vector(EC24Vector)

Constructs a vector.

```
[VB6]
void EC24Vector(
)
void EC24Vector(
    Long maxNumberOfElements
)
void EC24Vector(
    EC24Vector other
)
```

Parameters

maxNumberOfElements

Optionally, memory can be pre-allocated to accommodate a given number of elements.

other

EC24Vector object to be copied

EC24Vector.GetElement

Returns the vector element at the given index.

[VB6]

```
EC24 GetElement(  
    Long index  
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded) of the element to be accessed.

Remarks

If the given index is outside the bounds of the vector, the error code [EError_Parameter1OutOfRange](#) is set.

EC24Vector.operator[]

Gives access to the vector element at the given index.

[VB6]

```
EC24 operator[](  
    Long index  
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded) of the element to be accessed.

EC24Vector.operator=

Copies all the data from another EC24Vector object into the current EC24Vector object

[VB6]

```
EC24Vector operator=(  
    EC24Vector other  
)
```

Parameters

other

EC24Vector object to be copied

EC24Vector.RawDataPtr

Pointer to the vector data.

[VB6]

```
RawDataPtr As Long  
read-only
```

EC24Vector.SetElement

Modifies the vector element at the given index by the given value.

[VB6]

```
void SetElement(
    Long index,
    EC24 value
)
```

Parameters*index*Index, between **0** and [EVector::NumElements](#) (excluded), of the element to be modified.*value*

The new value for the element.

Remarks

If the given index is outside the bounds of the vector, the error code [EError_Parameter1OutOfRange](#) is set.

ECannyEdgeDetector Class

Manages a complete context for the Canny edge detector.

Remarks

The Canny edge detector operates on a grayscale BW8 image and delivers a black-and-white BW8 image where pixels have only 2 possible values: 0 and 255. Pixels corresponding to edges in the source image are set to value 255 in the output image; The other pixels are set to value 0.

PROPERTIES

HighThreshold

Sets the high hysteresis threshold for a pixel to be considered as an edge.

LowThreshold

Sets the low hysteresis threshold for a pixel to be considered as an edge.

SmoothingScale

The scale of the features of interest.

ThresholdingMode

M Sets the mode of the hysteresis thresholding.

THODS**Apply**

Apply the Canny edge detector on an image/ROI.

ECannyEdgeDetector

Constructs a ECannyEdgeDetector object initialized to its default values.

ResetSmoothingScale

E Prevents the smoothing of the source image by a Gaussian filter.

C

cannyEdgeDetector.Apply

Apply the Canny edge detector on an image/ROI.

[VB6]

```
void Apply(
    EROIBW8 source,
    EROIBW8 result
)
```

Parameters

source

The source image/ROI.

result

The output image/ROI.

Remarks

The output ROI must have the same size than the input ROI.

ECannyEdgeDetector.ECannyEdgeDetector

Constructs a ECannyEdgeDetector object initialized to its default values.

```
[VB6]  
  
void ECannyEdgeDetector(  
    ECannyEdgeDetector other  
)  
  
void ECannyEdgeDetector(  
)
```

Parameters

other

-

ECannyEdgeDetector.HighThreshold

Sets the high hysteresis threshold for a pixel to be considered as an edge.

```
[VB6]  
  
HighThreshold As Single  
  
read-write
```

ECannyEdgeDetector.LowThreshold

Sets the low hysteresis threshold for a pixel to be considered as an edge.

[VB6]

LowThreshold As Single

read-write

ECannyEdgeDetector.ResetSmoothingScale

Prevents the smoothing of the source image by a Gaussian filter.

[VB6]

```
void ResetSmoothingScale()  
)
```

Remarks

Calling this method is equivalent to set [ECannyEdgeDetector::SmoothingScale](#) to zero. It disables the use of the Gaussian filter.

ECannyEdgeDetector.SmoothingScale

The scale of the features of interest.

[VB6]

SmoothingScale As Single

read-write

Remarks

This scale corresponds to the standard deviation of the Gaussian filter that is used to smooth the source image before the computation of the gradient, hereby selecting the scale of the features of interest. If this scale is set to zero, no smoothing is achieved: The gradient is computed directly on the raw source image, speeding up the detector, but making the process much less reliable.

ECannyEdgeDetector.ThresholdingMode

Sets the mode of the hysteresis thresholding.

[VB6]

ThresholdingMode As ECannyThresholdingMode

read-write

Remarks

If the threshold mode is set to [ECannyThresholdingMode_Absolute](#), the threshold values are interpreted as absolute thresholds. In this case, the thresholds must be strictly positive real values.

If the threshold mode is set to [ECannyThresholdingMode_Relative](#), the thresholds are expressed as a fraction ranging from 0 to 1 of the maximum value of the gradient of the source image.

In either case, the low threshold must be less than the high threshold.

EChecker Class

Manages a complete context for the inspection tool based on image comparison in EasyOCV.

PROPERTIES

Average	Global intensity of the mother image.
DarkGray	-
DegreesOfFreedom	Boolean combination of EDegreesOfFreedom members, that indicates which degrees of freedom are to be considered.
Deviation	Global contrast of the mother image.
High	High threshold image for the adaptive segmentation.
HitHandle	Handle currently hit.
HitRoi	ROI currently hit.
LightGray	-
Low	Low threshold image for the adaptive segmentation.
Normalize	Current normalization mode.

NumAverageSamples	Number of samples that were accumulated in the "average" phase of the training.
NumDeviationSamples	Number of samples that were accumulated in the "deviation" phase of the training.
PanX	Current horizontal panning factor for use in display operations.
PanY	Current vertical panning factor for use in display operations.
Registered	Represents the source image, after it has been aligned with the reference image.
RelativeTolerance	Current tolerance factor to be used for threshold image setup.
ToleranceX	Current horizontal search tolerance, in pixels.
ToleranceY	Current vertical search tolerance, in pixels.
ZoomX	Current horizontal zooming factor for use in display operations.

ZoomY	M Current vertical zooming factor for use in display operations.
THODS	E
AddPathName	Adds a single file pathname.
Attach	Associates a source image to a checker context.
BatchLearn	Performs the learning sequence using the specified list of image files.
Drag	Moves the relevant ROI by means of its handle.
Draw	Draws one of the geometric items that define the EChecker tool.
DrawWithCurrentPen	Draws one of the geometric items that define the EChecker tool.
EChecker	Constructs an uninitialized checker context.
EmptyPathNames	Clears the list of file pathnames.
HitTest	Returns TRUE if the cursor is over one of the dragging handles.

Learn	Accumulates a reference image, following a sequence of operations.
Load	-
Register	Realigns and normalizes the source image.
Save	-
SetPan	Sets the panning factor for use in display operations.
SetTolerance	Sets the search margin, i.e. the width and height of the search area surrounding the alignment pattern (s).
SetZoom	Sets the zooming factor for use in display operations.

hecker.AddPathName

Adds a single file pathname.

[VB6]

```
void AddPathName(  
    String pathName  
)
```

Parameters

pathName

NULL terminated text string containing the file pathname.

EChecker.Attach

Associates a source image to a checker context.

```
[VB6]  
void Attach(  
    EROIBW8 source  
)
```

Parameters

source

Pointer to the source image.

Remarks

The source image is used in all consecutive learning/inspection operations.

EChecker.Average

Global intensity of the mother image.

```
[VB6]  
Average As Single
```

read-only

Remarks

Valid in mode [ENormalizationMode_Moments](#) only.

EChecker.BatchLearn

Performs the learning sequence using the specified list of image files.

[VB6]

```
void BatchLearn(  
    ELearningMode mode  
)
```

Parameters

mode

[ELearningMode_RmsDeviation](#) or [ELearningMode_AbsDeviation](#), depending on the preferred method of computing the deviations.

EChecker.DarkGray

-

[VB6]

```
DarkGray As Single
```

read-write

EChecker.DegreesOfFreedom

Boolean combination of [EDegreesOfFreedom](#) members, that indicates which degrees of freedom are to be considered.

[VB6]

DegreesOfFreedom As Long

read-write

EChecker.Deviation

Global contrast of the mother image.

[VB6]

Deviation As Single

read-only

Remarks

Valid in mode [ENormalizationMode_Moments](#) only.

EChecker.Drag

Moves the relevant ROI by means of its handle.

[VB6]

```
void Drag(
    Long x,
    Long y
)
```

Parameters

x

New horizontal cursor position.

y

New vertical cursor position.

EChecker.Draw

Draws one of the geometric items that define the [EChecker](#) tool.

[VB6]

```
void Draw(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean handles,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void Draw(
    Long graphicContext,
    ERGBColor color,
    EDrawingMode drawingMode,
    Boolean handles,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

```
void Draw(
    EDrawAdapter graphicContext,
    EDrawingMode drawingMode,
    Boolean handles,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

graphicContext

Device context of the drawing window.

drawingMode

ROI to be drawn, as defined by [EDrawingMode](#).

handles

TRUE if the dragging handles must be displayed.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the image is displayed in true scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a **0** value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

color

The color in which to draw the overlay.

EChecker.DrawWithCurrentPen

Draws one of the geometric items that define the [EChecker](#) tool.

[VB6]

```
void DrawWithCurrentPen(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean handles,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

graphicContext

Device context of the drawing window.

drawingMode

ROI to be drawn, as defined by [EDrawingMode](#).

handles

TRUE if the dragging handles must be displayed.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the image is displayed in true scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a **0** value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

EChecker.EChecker

Constructs an uninitialized checker context.

[VB6]

```
void EChecker(
    EChecker other
)
void EChecker(
)
```

Parameters

other

-

EChecker.EmptyPathNames

Clears the list of file pathnames.

[VB6]

```
void EmptyPathNames(
)
```

EChecker.High

High threshold image for the adaptive segmentation.

[VB6]

High As EIImageBW8

read-only

EChecker.HitHandle

Handle currently hit.

[VB6]

HitHandle As EDragHandle

read-only

EChecker.HitRoi

ROI currently hit.

[VB6]

HitRoi As ERoiHit

read-only

EChecker.HitTest

Returns **TRUE** if the cursor is over one of the dragging handles.

[VB6]

```
Boolean HitTest(  
    Long x,  
    Long y  
)
```

Parameters

x

Current horizontal cursor position.

X

Current vertical cursor position.

Remarks

In this case, [EChecker::HitRoi](#) returns the name of the ROI that has been hit, and [EChecker::HitHandle](#) returns the name of the corresponding handle.

EChecker.Learn

Accumulates a reference image, following a sequence of operations.

[VB6]

```
void Learn(  
    ELearningMode mode  
)
```

Parameters

mode

Current mode of operation in the learning sequence, as defined by [ELearningMode](#).

Remarks

First the model is reset; then the matching patterns are shown; next a series of images is presented to estimate the average gray levels; then a second series of images is presented to estimate the gray-level variations; finally, the threshold images are generated. A typical sequence with three reference images goes as follows: For standard deviation estimation `EChecker.Learn(ELearningMode_Reset)`; initializes. `EChecker.Register()`; realigns and normalizes 1st source image. `EChecker.Learn(ELearningMode_RmsDeviation)`; processes 1st image for deviation info. `EChecker.Register()`; realigns and normalizes 2nd source image. `EChecker.Learn(ELearningMode_RmsDeviation)`; processes 2nd image for deviation info. `EChecker.Register()`; realigns and normalizes 3rd source image. `EChecker.Learn(ELearningMode_RmsDeviation)`; processes 3rd image for deviation info. For robust deviation estimation `EChecker.Learn(ELearningMode_Reset)`; initializes. `EChecker.Register()`; realigns and normalizes 1st source image. `EChecker.Learn(ELearningMode_Average)`; processes 1st image for average info. `EChecker.Register()`; realigns and normalizes 2nd source image. `EChecker.Learn(ELearningMode_Average)`; processes 2nd image for average info. `EChecker.Register()`; realigns and normalizes 3rd source image. `EChecker.Learn(ELearningMode_Average)`; processes 3rd image for average info. `EChecker.Register()`; realigns and normalizes 1st source image. `EChecker.Learn(ELearningMode_RmsDeviation)`; processes 1st image for deviation info. `EChecker.Register()`; realigns and normalizes 2nd source image. `EChecker.Learn(ELearningMode_RmsDeviation)`; processes 2nd image for deviation info. `EChecker.Register()`; realigns and normalizes 3rd source image. `EChecker.Learn(ELearningMode_RmsDeviation)`; processes 3rd image for deviation info. `EChecker.Learn(ELearningMode_Ready)`; computes the threshold images.

EChecker.LightGray

-

[VB6]

LightGray As Single

read-write

EChecker.Load

-

```
[VB6]  
void Load(  
    ESerializer serializer  
)  
  
void Load(  
    String stream  
)
```

Parameters

serializer

-

stream

-

EChecker.Low

Low threshold image for the adaptive segmentation.

```
[VB6]  
Low As EImageBW8  
read-only
```

EChecker.Normalize

Current normalization mode.

```
[VB6]  
Normalize As ENormalizationMode  
read-write
```

EChecker.NumAverageSamples

Number of samples that were accumulated in the "average" phase of the training.

[VB6]

NumAverageSamples As Long

read-only

EChecker.NumDeviationSamples

Number of samples that were accumulated in the "deviation" phase of the training.

[VB6]

NumDeviationSamples As Long

read-only

EChecker.PanX

Current horizontal panning factor for use in display operations.

[VB6]

PanX As Single

read-only

EChecker.PanY

Current vertical panning factor for use in display operations.

[VB6]

PanY As Single

read-only

EChecker.Register

Realigns and normalizes the source image.

[VB6]

```
void Register()  
)
```

Remarks

Only the inspected ROI is processed. The first time this function is called, the current pattern ROI are used to define the search patterns. After registration, public member [EChecker::Registered](#) contains the realigned, normalized contents of the inspected ROI.

EChecker.Registered

Represents the source image, after it has been aligned with the reference image.

[VB6]

Registered As EIImageBW8

read-only

EChecker.RelativeTolerance

Current tolerance factor to be used for threshold image setup.

[VB6]

RelativeTolerance As Single

read-write

EChecker.Save

-

[VB6]

```
void Save(
    ESerializer serializer
)
void Save(
    String stream
)
```

Parameters

serializer

-

stream

-

EChecker.SetPan

Sets the panning factor for use in display operations.

```
[VB6]  
void SetPan(  
    Single panX,  
    Single panY  
)
```

Parameters

panX

Horizontal panning factor.

panY

Vertical panning factor.

EChecker.SetTolerance

Sets the search margin, i.e. the width and height of the search area surrounding the alignment pattern(s).

```
[VB6]  
void SetTolerance(  
    Long toleranceX,  
    Long toleranceY  
)
```

Parameters

toleranceX

Horizontal search tolerance, in pixels.

toleranceY

Vertical search tolerance, in pixels.

EChecker.SetZoom

Sets the zooming factor for use in display operations.

```
[VB6]  
void SetZoom(  
    Single zoom  
)  
void SetZoom(  
    Single zoomX,  
    Single zoomY  
)
```

Parameters

zoom

Magnification factor for zooming in or out in the horizontal and vertical directions (isotropic scaling).

zoomX

Magnification factor for zooming in or out in the horizontal direction.

zoomY

Magnification factor for zooming in or out in the vertical direction.

EChecker.ToleranceX

Current horizontal search tolerance, in pixels.

```
[VB6]  
ToleranceX As Long  
read-only
```

EChecker.ToleranceY

Current vertical search tolerance, in pixels.

[VB6]

ToleranceY As Long

read-only

EChecker.ZoomX

Current horizontal zooming factor for use in display operations.

[VB6]

ZoomX As Single

read-only

EChecker.ZoomY

Current vertical zooming factor for use in display operations.

[VB6]

ZoomY As Single

read-only

ECircle Class

Represents a model of a circle (or arc) in EasyGauge.

Base Class: [EFrame](#)

PROPERTIES

Amplitude	Angular amplitude of the ECircle object.
Apex	Apex point coordinates of a ECircle object.
ApexAngle	Angular position at the apex of a ECircle object.
ArcLength	Circle arc length of a ECircle object.
Diameter	Diameter of a ECircle object.
Direct	Flag indicating whether positive angles correspond to a clockwise or an anticlockwise rotation.
DirectInternal	-
End	End point coordinates of a ECircle object.

EndAngle	Angular position of the end of a ECircle object.
Full	Flag indicating whether the ECircle object is a full circle or not.
Org	Origin point coordinates of a ECircle object.
OrgAngle	Angular position from where the ECircle object extents.
Radius	M Radius of a ECircle object. E
THODS	
CopyTo	Copies all the data of the current ECircle object into another ECircle object and returns it.
Distance	Returns the smallest distance between this ECircle object an another ECircle .
ECircle	Constructs a ECircle object.
GetDistanceBetweenLineAndCircle	Computes the distance between a line and a circle.
GetDistanceBetweenPointAndCircle	Computes the distance between a point and a circle.

[GetIntersectionOfCircles](#)

Computes the intersections between two circles.
Returns the number of intersections and stores the found intersections in the provided point parameters.

[GetIntersectionOfLineAndCircle](#)

Computes the intersections between a line and circle. Returns the number of intersections and stores the found intersections in the provided point parameters.

[GetPoint](#)

Returns the coordinates of a particular point specified by its location along the circle arc.

[GetProjectionOfPointOnCircle](#)

Computes the projection of a point on a circle.

[operator=](#)

Copies all the data from another ECircle object into the current ECircle object

[Serialize](#)

-

[SetFromCenterAndOrigin](#)

Sets the geometric parameters (nominal position and diameter, angular position and amplitude) of an ECircle object.

[SetFromOriginMiddleEnd](#)

Sets the geometric parameters (nominal position and diameter, angular position and amplitude) of an ECircle object.

ECircle.Amplitude

Angular amplitude of the ECircle object.

[VB6]

Amplitude As Single

read-write

Remarks

The default value is **360**. A ECircle object is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), the angular position from where it extents, its angular amplitude, and its outline tolerance. The sign of the rotation angle depends whether the field of view is calibrated or not. * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value. * When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

ECircle.Apex

Apex point coordinates of a ECircle object.

[VB6]

Apex As EPoint

read-only

ECircle.ApexAngle

Angular position at the apex of a ECircle object.

[VB6]

ApexAngle As Single

read-only

Remarks

A ECircle object is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), the angular position from where it extents, its angular amplitude, and its outline tolerance. The sign of the rotation angle depends whether the field of view is calibrated or not. * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value. * When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

ECircle.ArcLength

Circle arc length of a ECircle object.

[VB6]

ArcLength As Single

read-only

Remarks

A ECircle object is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), the angular position from where it extents, its angular amplitude, and its outline tolerance.

ECircle.CopyTo

Copies all the data of the current ECircle object into another ECircle object and returns it.

[VB6]

```
ECircle CopyTo(  
    ECircle other  
)
```

Parameters

other

Pointer to the ECircle object in which the current ECircle object data have to be copied.

Remarks

In case of a **NULL** pointer, a new ECircle object will be created and returned.

ECircle.Diameter

Diameter of a ECircle object.

[VB6]

Diameter As Single

read-write

Remarks

A ECircle object is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), the angular position from where it extents, its angular amplitude, and its outline tolerance. By default, the diameter is **100**, which means 100 pixels when the field of view is not calibrated, and 100 physical units in case of a calibrated field of view.

ECircle.Direct

Flag indicating whether positive angles correspond to a clockwise or an anticlockwise rotation.

[VB6]

Direct As Boolean

read-only

Remarks

TRUE (default) means that angles increase anticlockwisely in a direct coordinate system, and clockwise in an inverse coordinate system. * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. * When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards.

ECircle.DirectInternal

-

[VB6]

DirectInternal As Boolean

read-write

ECircle.Distance

Returns the smallest distance between this ECircle object an another [ECircle](#).

[VB6]

```
Single Distance(
    ECircle circle
)
```

Parameters

circle

The other circle

ECircle(ECircle)

Constructs a ECircle object.

```
[VB6]

void ECircle(
)

void ECircle(
    EPoint center,
    Single diameter,
    Single originAngle,
    Boolean direct
)

void ECircle(
    EPoint center,
    EPoint origin,
    Boolean direct
)

void ECircle(
    EPoint center,
    Single diameter,
    Single originAngle,
    Single amplitude
)

void ECircle(
    EPoint origin,
    EPoint middle,
    EPoint end,
    Boolean fullCircle
)

void ECircle(
    ECircle other
)
```

Parameters

center

Center coordinates of the circle at its nominal position. The default value is **(0,0)**.

diameter

Nominal diameter of the circle. The default value is **100**.

originAngle

Nominal angular origin of the circle. The default value is 0.

direct

TRUE (default) means that angles increase anticlockwisely in a direct coordinate system.

origin

Origin point coordinates of the circle.

amplitude

Nominal angular amplitude of the circle. The default value is **360**.

middle

Middle point coordinates of the circle.

end

End point coordinates of the circle.

fullCircle

TRUE (default) in case of a full turn circle. If **fullCircle** is **FALSE**, **origin** and **end** give the circle's amplitude.

other

Another ECircle object to be copied in the new ECircle object.

ECircle.End

End point coordinates of a ECircle object.

[VB6]

End As EPoint

read-only

ECircle.EndAngle

Angular position of the end of a ECircle object.

[VB6]

EndAngle As Single

read-only

Remarks

A ECircle object is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), the angular position from where it extents, its angular amplitude, and its outline tolerance. The sign of the rotation angle depends whether the field of view is calibrated or not. * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value. * When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

ECircle.Full

Flag indicating whether the ECircle object is a full circle or not.

[VB6]

Full As Boolean

read-only

Remarks

By default (**TRUE**), the ECircle object is a full circle.

ECircle.GetDistanceBetweenLineAndCircle

Computes the distance between a line and a circle.

[VB6]

```
Single GetDistanceBetweenLineAndCircle(
    ELine line,
    ECircle circle,
    Boolean limited
)
```

Parameters

line

The line.

circle

The circle.

limited

Indicates if the line and circle parameters should be considered as infinite lines and full circles or as a segments and arcs.

ECircle.GetDistanceBetweenPointAndCircle

Computes the distance between a point and a circle.

[VB6]

```
Single GetDistanceBetweenPointAndCircle(
    EPoint pt,
    ECircle circle,
    Boolean limited
)
```

Parameters

pt

The point.

circle

The circle.

limited

Indicates if the circle parameter should be considered as a full circle or as an arc.

ECircle.GetIntersectionOfCircles

Computes the intersections between two circles. Returns the number of intersections and stores the found intersections in the provided point parameters.

[VB6]

```
Long GetIntersectionOfCircles(  
    ECircle circle1,  
    ECircle circle2,  
    EPoint intersection1,  
    EPoint intersection2,  
    Boolean limited  
)
```

Parameters

circle1

The first circle

circle2

The second circle

intersection1

The first intersection

intersection2

The second intersection

limited

Indicates if the circle parameters should be considered as full circles or as arcs.

Remarks

The function returns the number of intersections found. It will return -1 if the two circles are overlapping.

ECircle.GetIntersectionOfLineAndCircle

Computes the intersections between a line and circle. Returns the number of intersections and stores the found intersections in the provided point parameters.

[VB6]

```
Long GetIntersectionOfLineAndCircle(
    ELine line,
    ECircle circle,
    EPoint intersection1,
    EPoint intersection2,
    Boolean limited
)
```

Parameters

line

The line

circle

The circle

intersection1

The first intersection

intersection2

The second intersection

limited

Indicates if the line and circle parameters should be considered as infinite lines and full circles or as a segments and arcs.

Remarks

The function returns the number of intersections found.

ECircle.GetPoint

Returns the coordinates of a particular point specified by its location along the circle arc.

[VB6]

```
EPoint GetPoint(
    Single fraction
)
```

Parameters

fraction

Point location expressed as a fraction of the circle arc (range **[-1, +1]**).

ECircle.GetProjectionOfPointOnCircle

Computes the projection of a point on a circle.

```
[VB6]
EPoint GetProjectionOfPointOnCircle(
    EPoint pt,
    ECircle circle
)
```

Parameters

pt
The point.
circle
The circle.

ECircle.operator=

Copies all the data from another ECircle object into the current ECircle object

```
[VB6]
ECircle operator=(
    ECircle other
)
```

Parameters

other
ECircle object to be copied

ECircle.Org

Origin point coordinates of a ECircle object.

[VB6]

Org As EPoint

read-only

ECircle.OrgAngle

Angular position from where the ECircle object extents.

[VB6]

OrgAngle As Single

read-only

Remarks

A ECircle object is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), the angular position from where it extents, its angular amplitude, and its outline tolerance. The sign of the rotation angle depends whether the field of view is calibrated or not. * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value. * When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

ECircle.Radius

Radius of a ECircle object.

[VB6]

Radius As Single

read-write

Remarks

A ECircle object is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), the angular position from where it extents, its angular amplitude, and its outline tolerance. By default, the radius is **50**, which means 50 pixels when the field of view is not calibrated, and 50 physical units in case of a calibrated field of view.

ECircle.Serialize

-

[VB6]

```
void Serialize(
    ESerializer serializer,
    Long un32FileVersion
)
```

Parameters

serializer

-

un32FileVersion

-

ECircle.SetFromCenterAndOrigin

Sets the geometric parameters (nominal position and diameter, angular position and amplitude) of an ECircle object.

[VB6]

```
void SetFromCenterAndOrigin(
    EPoint center,
    EPoint origin,
    Boolean direct
)
```

Parameters

center

Center coordinates of the circle at its nominal position. The default value is **(0,0)**.

origin

Origin point coordinates of the circle.

direct

TRUE (default) means that angles increase anticlockwisely in a direct coordinate system.

ECircle.SetFromOriginMiddleEnd

Sets the geometric parameters (nominal position and diameter, angular position and amplitude) of an ECircle object.

[VB6]

```
void SetFromOriginMiddleEnd(
    EPoint origin,
    EPoint middle,
    EPoint end,
    Boolean fullCircle
)
```

Parameters

origin

Origin point coordinates of the circle.

middle

Middle point coordinates of the circle.

end

End point coordinates of the circle.

fullCircle

TRUE (default) in case of a full turn circle. If **fullCircle** is **FALSE**, **origin** and **end** give the circle's amplitude.

ECircleGauge Class

Manages a circle fitting gauge.

Base Class: [ECircleShape](#)

PROPERTIES

Active

Sets the flag indicating whether the gauge is active or not.

AverageDistance

-

Circle

Sets the nominal position (center coordinates), diameter, angular origin and amplitude of the circle fitting gauge according to a known circle.

FilteringThreshold

-

HVConstraint

-

InnerFilteringEnabled

Getter method for the **GetInnerFilteringEnabled** property. This property is the flag indicating if the inner sampled point filtering is enabled (**TRUE**), or not.

InnerFilteringThreshold	Sampled point inner filtering threshold.
MeasuredCircle	Information pertaining to the fitted circle.
MinAmplitude	-
MinArea	-
NumFilteringPasses	-
NumMeasuredPoints	-
NumSamples	-
NumSkipRanges	-
NumValidSamples	-
RectangularSamplingArea	-
SamplingStep	-
Shape	-

Smoothing	-
Thickness	-
Threshold	-
Tolerance	Searching area half thickness of the circle fitting gauge.
TransitionChoice	-
TransitionIndex	-
TransitionType	-
Type	Shape type.
Valid	M E
THODS	
AddSkipRange	Adds an item to the set of skip ranges and returns the index of the newly added range.

CopyTo	Copies all the data of the current ECircleGauge object into another ECircleGauge object, and returns it.
DisableInnnerFiltering	Disables inner sampled point filtering.
Drag	Moves a handle to a new position and updates the position parameters of the gauge.
Draw	Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode .
DrawWithCurrentPen	Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode .
ECircleGauge	Constructs a circle measurement context.
GetMeasuredPeak	Returns information pertaining to the default derivative peak, along one of the sample paths of the gauge.
GetMeasuredPoint	Returns the coordinates of a sample point, measured along one of the sample paths of the gauge.

GetMinNumFitSamples	Returns the minimum number of samples required for fitting on each side of the shape.
GetSample	Allows to retrieve the sample points found along the circle.
GetSkipRange	Allows to retrieve the start and end values of the skip range corresponding to the given index, if it is valid (i.e. if it has previously been created by a call to the ECircleGauge::AddSkipRange method).
HitTest	Checks whether the cursor is positioned over a handle (TRUE) or not (FALSE).
Measure	Triggers the point location or the model fitting operation.
MeasureSample	Computes the sample points along the sample path whose index in the list is given by the pathIndex parameter.
MeasureWithoutFitting	Triggers the point location without circle fitting operation.
operator=	Copies all the data from another ECircleGauge object into the current ECircleGauge object
Plot	Draws the profile that is crossed by one of the sample paths of the gauge, as defined by EPlotItem .

PlotWithCurrentPen

Draws the profile that is crossed by one of the sample paths of the gauge, as defined by [EPlotItem](#).

Process

Triggers the process pertaining to a shape or gauge and all the daughter gauges attach to it.

RemoveAllSkipRanges

Removes all the skip ranges previously created by a call to [ECircleGauge::AddSkipRange](#).

RemoveSkipRange

After a call to [ECircleGauge::AddSkipRange](#), removes the skip range with the given index.

SetMinNumFitSamples

Sets the minimum number of samples required for fitting on each side of the shape.

C

circleGauge.Active

Sets the flag indicating whether the gauge is active or not.

[VB6]

Active As Boolean

read-write

Remarks

When complex gauging is required, several gauges can be grouped together. Applying [ECircleGauge::Process](#) to the mother gauge or shape triggers the measurement of the whole. Only the active gauges will participate in the process. By default, the gauge is active (**TRUE**).

ECircleGauge.AddSkipRange

Adds an item to the set of skip ranges and returns the index of the newly added range.

[VB6]

```
Long AddSkipRange(
    Long start,
    Long end
)
```

Parameters

start

Beginning of the skip range.

end

End of the skip range.

Remarks

The samples indices between start and end (including the boundaries) will be discarded during the measurement process. The [AddSkipRange](#) method allows to define skip ranges in an [ECircleGauge](#). This means that, at measure time, samples belonging to these ranges will not be taken into account. A sample may belong to more than one skip range; to be discarded, a sample has to pertain to at least one range. Moreover, the skip ranges are allowed to overlap one another. The range is allowed to be reversed (i.e. end is not required to be greater than start). Also, start and end are not required to reference valid indices at the time of the call (i.e. the range may lie outside of the current return value for [ECircleGauge::NumSamples](#)).

ECircleGauge.AverageDistance

[VB6]

AverageDistance As Single

read-only

ECircleGauge.Circle

Sets the nominal position (center coordinates), diameter, angular origin and amplitude of the circle fitting gauge according to a known circle.

[VB6]

Circle As ECircle

read-write

ECircleGauge.CopyTo

Copies all the data of the current ECircleGauge object into another ECircleGauge object, and returns it.

[VB6]

```
ECircleGauge CopyTo(
ECircleGauge other,
Boolean recursive
<>)
```

Parameters

other

Pointer to the ECircleGauge object in which the current ECircleGauge object data have to be copied.

recursive

TRUE if the children gauges have to be copied as well, **FALSE** otherwise.

Remarks

In case of a **NULL** pointer, a new ECircleGauge object will be created and returned.

ECircleGauge::DisableInnnerFiltering

Disables inner sampled point filtering.

[VB6]

```
void DisableInnerFiltering()  
 )
```

Remarks

The inner sampled point filtering is activated as soon as the corresponding **ECircleGauge::InnerFilteringThreshold** is set.

ECircleGauge::Drag

Moves a handle to a new position and updates the position parameters of the gauge.

[VB6]

```
void Drag(  
 Long x,  
 Long y  
)
```

Parameters

x

Cursor current coordinates.

y

Cursor current coordinates.

ECircleGauge.Draw

Draws a graphical representation of a point location or model fitting gauge, as defined by [EDrawingMode](#).

[VB6]

```
void Draw(
    EDrawAdapter graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)

void Draw(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)

void Draw(
    Long graphicContext,
    ERGBColor color,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by [EDrawingMode](#).

daughters

TRUE if the daughters gauges are to be displayed also.

color

The color in which to draw the overlay.

ECircleGauge.DrawWithCurrentPen

Draws a graphical representation of a point location or model fitting gauge, as defined by [EDrawingMode](#).

[VB6]

```
void DrawWithCurrentPen(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by [EDrawingMode](#).

daughters

TRUE if the daughters gauges are to be displayed also.

ECircleGauge.ECircleGauge

Constructs a circle measurement context.

[VB6]

```
void ECircleGauge(
)
void ECircleGauge(
    ECircleGauge other
)
```

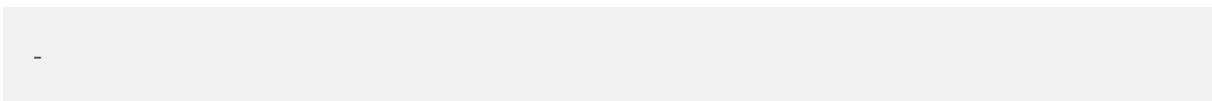
Parameters*other*

Another ECircleGauge object to be copied in the new ECircleGauge object.

Remarks

With the default constructor, all the parameters are initialized to their respective default values. With the copy constructor, the circle measurement context is based on a pre-existing ECircleGauge object. The gauge children are also copied. Hierarchy copying through a copy constructor is always recursive. To disable this recursion, use instead the [ECircleGauge::CopyTo](#) method.

ECircleGauge.FilteringThreshold

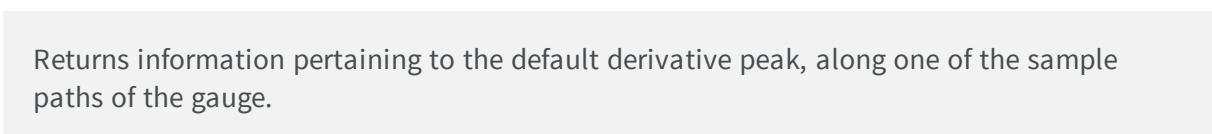


[VB6]

FilteringThreshold As Single

read-write

ECircleGauge.GetMeasuredPeak



Returns information pertaining to the default derivative peak, along one of the sample paths of the gauge.



[VB6]

```
EPeak GetMeasuredPeak (
Long index
 $\text{)}$ 
```

Parameters*index*

This argument must be left unchanged from its default value, i.e. **~0 (= 0xFFFFFFFF)**.

Remarks

[ECircleGauge::GetMeasuredPeak](#) returns the information about the derivative peak that meets the following two requirements: 1. It lies on the sample path inspected with the last call to [ECircleGauge::MeasureSample](#), and 1. It is associated with the edge-crossing point along the latter sample path that is selected by the transition choice parameter (cf. [ECircleGauge::TransitionChoice](#)).

Note. For this method to succeed, it is necessary to previously call [ECircleGauge::MeasureSample](#).

ECircleGauge.GetMeasuredPoint

Returns the coordinates of a sample point, measured along one of the sample paths of the gauge.

[VB6]

```
EPoint GetMeasuredPoint(
    Long index
)
```

Parameters

index

This argument must be left unchanged from its default value, i.e. **~0 (= 0xFFFFFFFF)**.

Remarks

These coordinates pertain to the world space; they are expressed in the reference frame to which the current ECircleGauge object belongs. The gauging process uses a list of sample points to find the shape position and size that best fit a given image. These sample points are measured along the sample paths defined by the gauge geometry.

[ECircleGauge::GetMeasuredPoint](#) returns the coordinates of the sample point that meets the following two requirements: 1. It lies on the sample path inspected with the last call to [ECircleGauge::MeasureSample](#), and 1. Among all the sample points along the latter sample path, it is the one selected by the [ECircleGauge::TransitionChoice](#) property.

Note. For this method to succeed, it is necessary to previously call [ECircleGauge::MeasureSample](#).

ECircleGauge.GetMinNumFitSamples

Returns the minimum number of samples required for fitting on each side of the shape.

```
[VB6]  
void GetMinNumFitSamples(  
    Long side0,  
    Long side1,  
    Long side2,  
    Long side3  
)
```

Parameters

side0

Minimum number of samples on the top side of the rectangle.

side1

Minimum number of samples on the left side of the rectangle.

side2

Minimum number of samples on the bottom side of the rectangle.

side3

Minimum number of samples on the right side of the rectangle.

Remarks

Irrelevant in case of a point location operation.

ECircleGauge.GetSample

Allows to retrieve the sample points found along the circle.

```
[VB6]
```

```
Boolean GetSample(
    EPoint pt,
    Long index
)
```

Parameters*pt*

EPoint structure to receive the position of the sample point.

index

The sample index

Remarks

The method provides the sample point corresponding to the supplied index. The returned value corresponds to the sample validity.

ECircleGauge.GetSkipRange

Allows to retrieve the start and end values of the skip range corresponding to the given index, if it is valid (i.e. if it has previously been created by a call to the [ECircleGauge::AddSkipRange](#) method).

[VB6]

```
void GetSkipRange(
    Long index,
    Long start,
    Long end
)
```

Parameters*index*

Index of the skip range.

start

Beginning of the skip range.

end

End of the skip range.

Remarks

Start is guaranteed to be smaller or equal to end.

ECircleGauge.HitTest

Checks whether the cursor is positioned over a handle (**TRUE**) or not (**FALSE**).

[VB6]

```
Boolean HitTest(  
    Boolean daughters  
)
```

Parameters

daughters

TRUE if the daughters gauges handles have to be considered as well.

ECircleGauge.HVConstraint

-

[VB6]

```
HVConstraint As Boolean  
read-write
```

ECircleGauge.InnerFilteringEnabled

Getter method for the **GetInnerFilteringEnabled** property. This property is the flag indicating if the inner sampled point filtering is enabled (**TRUE**), or not.

[VB6]

InnerFilteringEnabled As Boolean

read-only

Remarks

The inner sampled point filtering is activated as soon as the corresponding [ECircleGauge::InnerFilteringThreshold](#) is set. To disable inner filtering, use the [ECircleGauge::DisableInnerFiltering](#) method.

ECircleGauge.InnerFilteringThreshold

Sampled point inner filtering threshold.

[VB6]

InnerFilteringThreshold As Single

read-write

Remarks

If inner filtering is enabled, the sampled points that have been found inside the measured circle are filtered in regard of their distance to it. If this distance is greater than the threshold, the sampled point is set as invalid, and removed from the measure. This distance is in physical units. The inner sampled point filtering is activated as soon as the corresponding threshold is set. To disable inner filtering, use the [ECircleGauge::DisableInnerFiltering](#) method.

ECircleGauge.Measure

Triggers the point location or the model fitting operation.

[VB6]

```
void Measure(
    EROIBW8 sourceImage
)
```

Parameters

sourceImage

Pointer to the source image.

Remarks

When this method is called, and if not enough valid points are detected, then the method returns directly, and the measured gauge is set to the nominal parameters.

ECircleGauge.MeasuredCircle

Information pertaining to the fitted circle.

[VB6]

MeasuredCircle As ECircle

read-only

ECircleGauge.MeasureSample

Computes the sample points along the sample path whose index in the list is given by the **pathIndex** parameter.

[VB6]

```
void MeasureSample(
    EROIBW8 sourceImage,
    Long pathIndex
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

pathIndex

Sample path index.

Remarks

This method stores its results into a temporary variable inside the ECircleGauge object.

ECircleGauge.MeasureWithoutFitting

Triggers the point location without circle fitting operation.

[VB6]

```
void MeasureWithoutFitting(
    EROIBW8 sourceImage
)
```

Parameters

sourceImage

Source image.

Remarks

This method performs the actual measurement for each transition, but does not perform the circle fitting. This means that individual samples will be available through the [ECircleGauge::GetSample](#) method, but the gauge position will not be changed. Please note that the filtering will not be performed in this method, since it relies upon the fitting process. The filtering parameters will thus be unused.

ECircleGauge.MinAmplitude

[VB6]

MinAmplitude As Long

read-write

ECircleGauge.MinArea

-

[VB6]

MinArea As Long

read-write

ECircleGauge.NumFilteringPasses

-

[VB6]

NumFilteringPasses As Long

read-write

ECircleGauge.NumMeasuredPoints

-

[VB6]

NumMeasuredPoints As Long

read-only

ECircleGauge.NumSamples

-

[VB6]

NumSamples As Long

read-only

ECircleGauge.NumSkipRanges

-

[VB6]

NumSkipRanges As Long

read-only

ECircleGauge.NumValidSamples

-

[VB6]

NumValidSamples As Long

read-only

ECircleGauge.operator=

Copies all the data from another ECircleGauge object into the current ECircleGauge object

[VB6]

```
ECircleGauge operator=(  
    ECircleGauge other  
)
```

Parameters

other

ECircleGauge object to be copied

ECircleGauge.Plot

Draws the profile that is crossed by one of the sample paths of the gauge, as defined by [EPlotItem](#).

[VB6]

```
void Plot(
    EDrawAdapter graphicContext,
    EPlotItem drawItems,
    Single width,
    Single height,
    Single originX,
    Single originY
)

void Plot(
    Long graphicContext,
    EPlotItem drawItems,
    Single width,
    Single height,
    Single originX,
    Single originY
)

void Plot(
    Long graphicContext,
    ERGBColor color,
    EPlotItem drawItems,
    Single width,
    Single height,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawItems

Boolean combination of [EPlotItem](#) members, that indicates which items must be displayed.

width

Width of the plot.

height

Height of the plot.

originX

Origin point coordinates of the plot along the X axis.

originY

Origin point coordinates of the plot along the Y axis.

color

The color in which to draw the overlay.

Remarks

The sample path that is taken into considered is the one inspected with the last call to [ECircleGauge::MeasureSample](#).

Note. For this method to succeed, it is necessary to previously call [ECircleGauge::MeasureSample](#).

ECircleGauge.PlotWithCurrentPen

Draws the profile that is crossed by one of the sample paths of the gauge, as defined by [EPlotItem](#).

[VB6]

```
void PlotWithCurrentPen(
    Long graphicContext,
    EPlotItem drawItems,
    Single width,
    Single height,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawItems

Boolean combination of [EPlotItem](#) members, that indicates which items must be displayed.

width

Width of the plot.

height

Height of the plot.

originX

Origin point coordinates of the plot along the X axis.

originY

Origin point coordinates of the plot along the Y axis.

Remarks

The sample path that is taken into considered is the one inspected with the last call to [ECircleGauge::MeasureSample](#).

Note. For this method to succeed, it is necessary to previously call [ECircleGauge::MeasureSample](#).

ECircleGauge.Process

Triggers the process pertaining to a shape or gauge and all the daughter gauges attach to it.

[VB6]

```
void Process(
    EROIBW8 sourceImage,
    Boolean daughters
)
```

Parameters

sourceImage

Pointer to the source image.

daughters

Flag indicating whether the daughters shapes inherit of the same behavior.

Remarks

When complex gauging is required, several gauges can be grouped together. Applying **Process** to the mother gauge or shape triggers the measurement of the whole. Only the active gauges will participate in the process.

ECircleGauge.RectangularSamplingArea

[VB6]

RectangularSamplingArea As Boolean

read-write

ECircleGauge.RemoveAllSkipRanges

Removes all the skip ranges previously created by a call to [ECircleGauge::AddSkipRange](#).

[VB6]

```
void RemoveAllSkipRanges()  
{}
```

ECircleGauge.RemoveSkipRange

After a call to [ECircleGauge::AddSkipRange](#), removes the skip range with the given index.

[VB6]

```
void RemoveSkipRange(  
    Long index  
)
```

Parameters

index

Index of the skip range to remove, as returned by [ECircleGauge::AddSkipRange](#).

ECircleGauge.SamplingStep

[VB6]

SamplingStep As Single

read-write

ECircleGauge.SetMinNumFitSamples

Sets the minimum number of samples required for fitting on each side of the shape.

[VB6]

```
void SetMinNumFitSamples(  
    Long side0,  
    Long side1,  
    Long side2,  
    Long side3  
)
```

Parameters

side0

Required number of samples to correctly fit the circle. The default value is **3**. It is the only parameter taken into account.

side1

Not used.

side2

Not used.

side3

Not used.

Remarks

Irrelevant in case of a point location operation. When the [ECircleGauge::Measure](#) method is called, and if not enough valid points are detected, then the method returns directly, and the measured gauge is set to the nominal parameters.

ECircleGauge.Shape

-

[VB6]

Shape As ECircle

read-only

ECircleGauge.Smoothing

-

[VB6]

Smoothing As Long

read-write

ECircleGauge.Thickness

-

[VB6]

Thickness As Long

read-write

ECircleGauge.Threshold

-

[VB6]

Threshold As Long

read-write

ECircleGauge.Tolerance

Searching area half thickness of the circle fitting gauge.

[VB6]

Tolerance As Single

read-write

Remarks

A circle fitting gauge is fully defined knowing its nominal position (given by the coordinates of its center), its nominal radius (diameter), the angular position from where it extents, its angular amplitude and its outline tolerance. By default, the searching area thickness of the circle fitting gauge is **20** (2x10), which means 20 pixels when the field of view is not calibrated, and 20 physical units in case of a calibrated field of view.

ECircleGauge.TransitionChoice

-

[VB6]

TransitionChoice As ETransitionChoice

read-write

ECircleGauge.TransitionIndex

-

[VB6]

TransitionIndex As Long

read-write

ECircleGauge.TransitionType

-

[VB6]

TransitionType As ETransitionType

read-write

ECircleGauge.Type

Shape type.

[VB6]

Type As EShapeType

read-only

ECircleGauge.Valid

-

[VB6]

Valid As Boolean

read-only

ECircleShape Class

-

Base Class: [EShape](#)

Derived Class(es): [ECircleGauge](#)

PROPERTIES

[Amplitude](#)

-

[Angle](#)

-

[Apex](#)

-

ApexAngle	-
ArcLength	-
Center	-
CenterX	-
CenterY	-
Circle	-
Diameter	-
Direct	-
End	-
EndAngle	-
Full	-
Org	-

OrgAngle	-
Radius	-
Scale	-
Type	M Shape type.
THODS	E
Closest	Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use EShape::ClosestShape .
CopyTo	-
Drag	-
Draw	Draws a graphical representation of a shape, as defined by EDrawingMode .
DrawWithCurrentPen	Draws a graphical representation of a shape, as defined by EDrawingMode .
GetPoint	Returns the coordinates of a particular point specified by its location along the circle arc.

[HitTest](#)

-

[operator=](#)

Copies all the data from another ECircleShape object into the current ECircleShape object

[SetCenterXY](#)

Sets the center coordinates of a ECircleShape object.

[SetFromCenterAndOrigin](#)

-

[SetFromOriginMiddleEnd](#)

E-

C

ircleShape.Amplitude

[VB6]

Amplitude As Single

read-write

ECircleShape.Angle

[VB6]

Angle As Single

read-write

ECircleShape.Apex

-

[VB6]

Apex As EPoint

read-only

ECircleShape.ApexAngle

-

[VB6]

ApexAngle As Single

read-only

ECircleShape.ArcLength

-

[VB6]

ArcLength As Single

read-only

ECircleShape.Center

-

[VB6]

Center As EPoint

read-write

ECircleShape.CenterX

-

[VB6]

CenterX As Single

read-only

ECircleShape.CenterY

-

[VB6]

CenterY As Single

read-only

ECircleShape.Circle

-

[VB6]

Circle As ECircle

read-write

ECircleShape.Closest

Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use [EShape::ClosestShape](#).

[VB6]

```
void Closest()  
)
```

ECircleShape.CopyTo

-

[VB6]

```
ECircleShape CopyTo(  
    ECircleShape dest,  
    Boolean bRecursive  
)
```

Parameters

dest

-

bRecursive

-

ECircleShape.Diameter

[VB6]

Diameter As Single

read-write

ECircleShape.Direct

[VB6]

Direct As Boolean

read-only

ECircleShape.Drag

-

[VB6]

```
void Drag(
    Long n32CursorX,
    Long n32CursorY
)
```

Parameters

n32CursorX

n32CursorY

ECircleShape.Draw

Draws a graphical representation of a shape, as defined by [EDrawingMode](#).

-

[VB6]

```
void Draw(
    EDrawAdapter graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)

void Draw(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

```
void Draw(
    Long graphicContext,
    ERGBColor color,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by [EDrawingMode](#).

daughters

TRUE if the daughters gauges are to be displayed also.

color

-

ECircleShape.DrawWithCurrentPen

Draws a graphical representation of a shape, as defined by [EDrawingMode](#).

[VB6]

```
void DrawWithCurrentPen(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by [EDrawingMode](#).

daughters

TRUE if the daughters gauges are to be displayed also.

ECircleShape.End

-

[VB6]

End As EPoint

read-only

ECircleShape.EndAngle

-

[VB6]

EndAngle As Single

read-only

ECircleShape.Full

-

[VB6]

Full As Boolean

read-only

ECircleShape.GetPoint

Returns the coordinates of a particular point specified by its location along the circle arc.

[VB6]

```
EPoint GetPoint(  
    Single fraction  
)
```

Parameters

fraction

Point location expressed as a fraction of the circle arc (range **[-1, +1]**).

ECircleShape.HitTest

-

[VB6]

```
Boolean HitTest(  
    Boolean bDaughters  
)
```

Parameters

bDaughters

ECircleShape.operator=

Copies all the data from another ECircleShape object into the current ECircleShape object

[VB6]

```
ECircleShape operator=(  
    ECircleShape other  
)
```

Parameters

other

ECircleShape object to be copied

ECircleShape.Org

-

[VB6]

Org As EPoint

read-only

ECircleShape.OrgAngle

-

[VB6]

OrgAngle As Single

read-only

ECircleShape.Radius

-

[VB6]

Radius As Single

read-write

ECircleShape.Scale

-

[VB6]

Scale As Single

read-write

ECircleShape.SetCenterXY

Sets the center coordinates of a ECircleShape object.

[VB6]

```
void SetCenterXY(  
    Single centerX,  
    Single centerY  
)
```

Parameters

centerX

Center coordinates of the ECircleShape object.

centerY

Center coordinates of the ECircleShape object.

ECircleShape.SetFromCenterAndOrigin

-

[VB6]

```
void SetFromCenterAndOrigin(
    EPoint center,
    EPoint origin,
    Boolean direct
)
```

Parameters

center

-

origin

-

direct

-

ECircleShape.SetFromOriginMiddleEnd

-

[VB6]

```
void SetFromOriginMiddleEnd(
    EPoint origin,
    EPoint middle,
    EPoint end,
    Boolean fullCircle
)
```

Parameters

origin

```
-  
middle  
  
-  
end  
  
-  
fullCircle  
-
```

ECircleShape.Type

Shape type.

[VB6]

Type As EShapeType

read-only

ECodedElement Class

This class encapsulates either an object or a hole in an object, in a coded image.

Remarks

This abstract class provides a large set of methods applicable to a particular coded element. The set includes methods to get the features of a coded element, to draw coded elements, and to render flexible masks.

Derived Class(es): [EObject](#) [EHook](#)

PROPERTIES

Area

Returns the number of pixels inside the coded element.

[BottomLimit](#)

Returns the highest (integer) Y-coordinate of all the pixels of the coded element.

[BoundingBox](#)

Returns the bounding box of the coded element (Feret box at orientation 0°).

[BoundingBoxCenter](#)

Returns the coordinates of the center of the bounding box of the coded element.

[BoundingBoxCenterX](#)

Returns the abscissa of the center of the bounding box of the coded element.

[BoundingBoxCenterY](#)

Returns the ordinate of the center of the bounding box of the coded element.

[BoundingBoxHeight](#)

Returns the height of the bounding box (Feret diameter at 90°).

[BoundingBoxWidth](#)

Returns the width of the bounding box (Feret diameter at 0°).

[Contour](#)

Returns the coordinates of the starting point of the countour of the coded element.

ContourX	Returns the abscissa of the starting point of the countour of the coded element.
ContourY	Returns the ordinate of the starting point of the countour of the coded element.
Eccentricity	Returns the eccentricity of the ellipse of inertia.
ElementIndex	Returns the index of the coded element.
EllipseAngle	Returns the angle of the ellipse of inertia.
EllipseHeight	Returns the length of the short axis of the ellipse of inertia.
EllipseWidth	Returns the length of the long axis of the ellipse of inertia.
FeretBox22Box	Returns the Feret box at orientation 22.5°.
FeretBox22Center	Returns the coordinates of the center of the Feret box oriented at 22.5°.
FeretBox22CenterX	Returns the abscissa of the center of the Feret box oriented at 22.5°.

[FeretBox22CenterY](#)

Returns the ordinate of the center of the Feret box oriented at 22.5°.

[FeretBox22Height](#)

Returns the height of the Feret box oriented at 22.5° (Feret diameter at 112.5°).

[FeretBox22Width](#)

Returns the width of the Feret box oriented at 22.5° (Feret diameter at 22.5°).

[FeretBox45Box](#)

Returns the Feret box at orientation 45°.

[FeretBox45Center](#)

Returns the coordinates of the center of the Feret box oriented at 45°.

[FeretBox45CenterX](#)

Returns the abscissa of the center of the Feret box oriented at 45°.

[FeretBox45CenterY](#)

Returns the ordinate of the center of the Feret box oriented at 45°.

[FeretBox45Height](#)

Returns the height of the Feret box oriented at 45° (Feret diameter at 135°).

[FeretBox45Width](#)

Returns the width of the Feret box oriented at 45° (Feret diameter at 45°).

FeretBox68Box	Returns the Feret box at orientation 67.5°.
FeretBox68Center	Returns the coordinates of the center of the Feret box oriented at 67.5°.
FeretBox68CenterX	Returns the abscissa of the center of the Feret box oriented at 67.5°.
FeretBox68CenterY	Returns the ordinate of the center of the Feret box oriented at 67.5°.
FeretBox68Height	Returns the height of the Feret box oriented at 67.5° (Feret diameter at 157.5°).
FeretBox68Width	Returns the width of the Feret box oriented at 67.5° (Feret diameter at 67.5°).
GravityCenter	Returns the gravity center of the coded element.
GravityCenterX	Returns the abscissa of the gravity center of the coded element.
GravityCenterY	Returns the ordinate of the gravity center of the coded element.

IsCodedElement	Tests whether the coded element is an object, a hole or a coded element.
IsHole	Tests whether the coded element is an object, a hole or a coded element.
IsObject	Tests whether the coded element is an object, a hole or a coded element.
LargestRun	Returns the length of the largest run inside the coded element.
LayerIndex	Returns the index of the layer in the coded image to which the coded element belongs.
LeftLimit	Returns the lowest (integer) X-coordinate of all the pixels of the coded element.
MinimumEnclosingRectangle	Returns the Minimum-Area Enclosing Rectangle.
MinimumEnclosingRectangleAngle	Returns the angle of the Minimum-Area Enclosing Rectangle.
MinimumEnclosingRectangleCenter	Returns the coordinates of the center of the Minimum-Area Enclosing Rectangle.
MinimumEnclosingRectangleCenter X	Returns the abscissa of the center of the Minimum-Area Enclosing Rectangle.

MinimumEnclosingRectangleCenterY	Returns the ordinate of the center of the Minimum-Area Enclosing Rectangle.
MinimumEnclosingRectangleHeight	Returns the height of the Minimum-Area Enclosing Rectangle.
MinimumEnclosingRectangleWidth	Returns the width of the Minimum-Area Enclosing Rectangle.
RightLimit	Returns the highest (integer) X-coordinate of all the pixels of the coded element.
RunCount	Returns the number of runs inside the coded element.
RunsIterator	Returns an iterator to the runs of the coded element.
SigmaX	Returns the centered moment of inertia around X (average squared X-deviation).
SigmaXX	Returns the centered cross moment of inertia (average X-deviation * Y-deviation).
SigmaXY	Returns the reduced, centered moment of inertia (around the principal inertia axis).

SigmaY	Returns the centered moment of inertia around Y (average squared Y-deviation).
SigmaYY	Returns the reduced, centered moment of inertia (around the secondary inertia axis).
TopLimit	M Returns the lowest (integer) Y-coordinate of all the pixels of the coded element.
THODS	E
AsHole	Down-casts the coded element as a hole.
AsObject	Down-casts the coded element as an object.
ComputeConvexHull	Computes the convex hull of the coded element.
ComputeFeretBox	Computes the Feret box at a specific orientation.
ComputePixelGrayAverage	Computes the average gray-level value of the pixels of a given image over the coded element.
ComputePixelGrayDeviation	Computes the standard deviation of the gray-level values of a given image over the coded element.

[ComputePixelGrayVariance](#)

Computes the variance of the gray-level values of a given image over the coded element.

[ComputePixelMax](#)

Computes the maximum gray level of the pixels of a given image over the coded element.

[ComputePixelMin](#)

Computes the minimum gray level of the pixels of a given image over the coded element.

[ComputeWeightedGravityCenter](#)

Computes the gravity center of a given image over the coded element.

[GetCentralMoment](#)

Computes the central, two-dimensional moment of order (p,q).

[GetMoment](#)

Computes the raw, two-dimensional moment of order (p,q).

[GetNormalizedCentralMoment](#)

Computes the scale-invariant, central, two-dimensional moment of order (p,q).

[RenderMask](#)

Creates a Flexible Mask from the coded element.

ECodedElement.Area

Returns the number of pixels inside the coded element.

[VB6]

Area As Long

read-only

Remarks

Equivalently, the area corresponds to the sum of the length of the runs of the coded element.

ECodedElement.AsHole

Down-casts the coded element as a hole.

[VB6]

EHole AsHole(
 `)`

Remarks

This method throws an exception if the coded element is in fact an object.

ECodedElement.AsObject

Down-casts the coded element as an object.

[VB6]

```
EObject AsObject(
    )
```

Remarks

This method throws an exception if the coded element is in fact a hole.

EcodedElement.BottomLimit

Returns the highest (integer) Y-coordinate of all the pixels of the coded element.

[VB6]

BottomLimit As Long

read-only

Remarks

For a coded element E, this value is defined as:
$$\max\{y \mid \exists x \quad (x, y \in E) \quad \max(y \mid \exists x \quad (x, y \in E)) \leq y \leq \max(y \mid \exists x \quad (x, y \in E))\}$$

EcodedElement.BoundingBox

Returns the bounding box of the coded element (Feret box at orientation 0°).

[VB6]

BoundingBox As ERotatedBoundingBox

read-only

ECodedElement.BoundingBoxCenter

Returns the coordinates of the center of the bounding box of the coded element.

[VB6]

BoundingBoxCenter As EPoint

read-only

ECodedElement.BoundingBoxCenterX

Returns the abscissa of the center of the bounding box of the coded element.

[VB6]

BoundingBoxCenterX As Single

read-only

ECodedElement.BoundingBoxCenterY

Returns the ordinate of the center of the bounding box of the coded element.

[VB6]

BoundingBoxCenterY As Single

read-only

ECodedElement.BoundingBoxHeight

Returns the height of the bounding box (Feret diameter at 90°).

[VB6]

BoundingBoxHeight As Single

read-only

ECodedElement.BoundingBoxWidth

Returns the width of the bounding box (Feret diameter at 0°).

[VB6]

BoundingBoxWidth As Single

read-only

ECodedElement.ComputeConvexHull

Computes the convex hull of the coded element.

[VB6]

```
void ComputeConvexHull(  
    EPathVector result  
)
```

Parameters

result

The output vector where to store the convex hull.

ECodedElement.ComputeFeretBox

Computes the Feret box at a specific orientation.

[VB6]

```
ERotatedBoundingBox ComputeFeretBox(
    Single angle
)
```

Parameters

angle

The orientation of interest (in the current angle units).

ECodedElement.ComputePixelGrayAverage

Computes the average gray-level value of the pixels of a given image over the coded element.

[VB6]

```
Single ComputePixelGrayAverage(
    EROIBW8 image
)
```

Parameters

image

The input image.

EcodedElement.ComputePixelGrayDeviation

Computes the standard deviation of the gray-level values of a given image over the coded element.

[VB6]

```
Single ComputePixelGrayDeviation(
    EROIBW8 image
)
```

Parameters

image

The input image.

EcodedElement.ComputePixelGrayVariance

Computes the variance of the gray-level values of a given image over the coded element.

[VB6]

```
Double ComputePixelGrayVariance(
    EROIBW8 image
)
```

Parameters

image

The input image.

EcodedElement.ComputePixelMax

Computes the maximum gray level of the pixels of a given image over the coded element.

[VB6]

```
EBW8 ComputePixelMax(  
    EROIBW8 image  
)
```

Parameters

image

The input image.

ECodedElement.ComputePixelMin

Computes the minimum gray level of the pixels of a given image over the coded element.

[VB6]

```
EBW8 ComputePixelMin(  
    EROIBW8 image  
)
```

Parameters

image

The input image.

ECodedElement.ComputeWeightedGravityCenter

Computes the gravity center of a given image over the coded element.

[VB6]

```
EPoint ComputeWeightedGravityCenter(  
    EROIBW8 image  
)
```

Parameters

image

The input image.

ECodedElement.Contour

Returns the coordinates of the starting point of the countour of the coded element.

[VB6]

Contour As EPoint

read-only

Remarks

More precisely, the leftmost pixel over the topmost row of the coded element is taken into consideration.

ECodedElement.ContourX

Returns the abscissa of the starting point of the countour of the coded element.

[VB6]

ContourX As Long

read-only

ECodedElement.ContourY

Returns the ordinate of the starting point of the countour of the coded element.

[VB6]

ContourY As Long

read-only

ECodedElement.Eccentricity

Returns the eccentricity of the ellipse of inertia.

[VB6]

Eccentricity As Single

read-only

Remarks

The ellipse of inertia is defined as the ellipse that has the same second order moments as the original coded element. The eccentricity is zero for circular objects and one for a line-shaped objects.

ECodedElement.ElementIndex

Returns the index of the coded element.

[VB6]

ElementIndex As Long

read-only

Remarks

If the coded element is an object, its index is relative to the layer to which it belongs. If the coded element is a hole, its index is relative to its parent object.

ECodedElement.EllipseAngle

Returns the angle of the ellipse of inertia.

[VB6]

EllipseAngle As Single

read-only

Remarks

The ellipse of inertia is defined as the ellipse that has the same second order moments as the original coded element.

ECodedElement.EllipseHeight

Returns the length of the short axis of the ellipse of inertia.

[VB6]

EllipseHeight As Single

read-only

Remarks

The ellipse of inertia is defined as the ellipse that has the same second order moments as the original coded element.

ECodedElement.EllipseWidth

Returns the length of the long axis of the ellipse of inertia.

[VB6]

EllipseWidth As Single

read-only

Remarks

The ellipse of inertia is defined as the ellipse that has the same second order moments as the original coded element.

ECodedElement.FeretBox22Box

Returns the Feret box at orientation 22.5°.

[VB6]

FeretBox22Box As ERotatedBoundingBox

read-only

ECodedElement.FeretBox22Center

Returns the coordinates of the center of the Feret box oriented at 22.5°.

[VB6]

FeretBox22Center As EPoint

read-only

ECodedElement.FeretBox22CenterX

Returns the abscissa of the center of the Feret box oriented at 22.5°.

[VB6]

FeretBox22CenterX As Single

read-only

ECodedElement.FeretBox22CenterY

Returns the ordinate of the center of the Feret box oriented at 22.5°.

[VB6]

FeretBox22CenterY As Single

read-only

ECodedElement.FeretBox22Height

Returns the height of the Feret box oriented at 22.5° (Feret diameter at 112.5°).

[VB6]

FeretBox22Height As Single

read-only

ECodedElement.FeretBox22Width

Returns the width of the Feret box oriented at 22.5° (Feret diameter at 22.5°).

[VB6]

FeretBox22Width As Single

read-only

ECodedElement.FeretBox45Box

Returns the Feret box at orientation 45°.

[VB6]

FeretBox45Box As ERotatedBoundingBox

read-only

ECodedElement.FeretBox45Center

Returns the coordinates of the center of the Feret box oriented at 45°.

[VB6]

FeretBox45Center As EPoint

read-only

ECodedElement.FeretBox45CenterX

Returns the abscissa of the center of the Feret box oriented at 45°.

[VB6]

FeretBox45CenterX As Single

read-only

ECodedElement.FeretBox45CenterY

Returns the ordinate of the center of the Feret box oriented at 45°.

[VB6]

FeretBox45CenterY As Single

read-only

ECodedElement.FeretBox45Height

Returns the height of the Feret box oriented at 45° (Feret diameter at 135°).

[VB6]

FeretBox45Height As Single

read-only

ECodedElement.FeretBox45Width

Returns the width of the Feret box oriented at 45° (Feret diameter at 45°).

[VB6]

FeretBox45Width As Single

read-only

ECodedElement.FeretBox68Box

Returns the Feret box at orientation 67.5°.

[VB6]

FeretBox68Box As ERotatedBoundingBox

read-only

ECodedElement.FeretBox68Center

Returns the coordinates of the center of the Feret box oriented at 67.5°.

[VB6]

FeretBox68Center As EPoint

read-only

ECodedElement.FeretBox68CenterX

Returns the abscissa of the center of the Feret box oriented at 67.5°.

[VB6]

FeretBox68CenterX As Single

read-only

ECodedElement.FeretBox68CenterY

Returns the ordinate of the center of the Feret box oriented at 67.5°.

[VB6]

FeretBox68CenterY As Single

read-only

ECodedElement.FeretBox68Height

Returns the height of the Feret box oriented at 67.5° (Feret diameter at 157.5°).

[VB6]

FeretBox68Height As Single

read-only

ECodedElement.FeretBox68Width

Returns the width of the Feret box oriented at 67.5° (Feret diameter at 67.5°).

[VB6]

FeretBox68Width As Single

read-only

ECodedElement.GetCentralMoment

Computes the central, two-dimensional moment of order (p,q).

[VB6]

```
Single GetCentralMoment(
  Long p,
  Long q
)
```

Parameters

p

Order of the moment along the X-axis.

q

Order of the moment along the Y-axis.

Remarks

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q I(x, y)$$

ECodedElement.GetMoment

Computes the raw, two-dimensional moment of order (p,q).

[VB6]

```
Double GetMoment(
    Long p,
    Long q
)
```

Parameters*p*

Order of the moment along the X-axis.

q

Order of the moment along the Y-axis.

Remarks

$$M_{pq} = \sum_x \sum_y x^p y^q I(x, y)$$

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\frac{p+q}{2}}}$$

EEncodedElement.GetNormalizedCentralMoment

Computes the scale-invariant, central, two-dimensional moment of order (p,q).

[VB6]

```
Single GetNormalizedCentralMoment(
    Long p,
    Long q
)
```

Parameters*p*

Order of the moment along the X-axis.

q

Order of the moment along the Y-axis.

Remarks

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\frac{p+q}{2}}}$$

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\frac{p+q}{2}}}$$

ECodedElement.GravityCenter

Returns the gravity center of the coded element.

[VB6]

GravityCenter As EPoint

read-only

ECodedElement.GravityCenterX

Returns the abscissa of the gravity center of the coded element.

[VB6]

GravityCenterX As Single

read-only

Remarks

For a coded element E, this value is defined as:
$$\frac{\sum_{(x,y) \in E} x}{\sum_{(x,y) \in E} 1}$$

$$\frac{\sum_{(x,y) \in E} x}{\sum_{(x,y) \in E} 1}$$

ECodedElement.GravityCenterY

Returns the ordinate of the gravity center of the coded element.

[VB6]

GravityCenterY As Single

read-only

Remarks

For a coded element E, this value is defined as:
$$\frac{\sum_{(x,y) \in E} y}{\sum_{(x,y) \in E} 1}$$

$$\frac{\sum_{(x,y) \in E} y}{\sum_{(x,y) \in E} 1}$$

ECodedElement.IsCodedElement

Tests whether the coded element is an object, a hole or a coded element.

[VB6]

IsCodedElement As Boolean

read-only

ECodedElement.IsHole

Tests whether the coded element is an object, a hole or a coded element.

[VB6]

IsHole As Boolean

read-only

ECodedElement.IsObject

Tests whether the coded element is an object, a hole or a coded element.

[VB6]

IsObject As Boolean

read-only

ECodedElement.LargestRun

Returns the length of the largest run inside the coded element.

[VB6]

LargestRun As Long

read-only

ECodedElement.LayerIndex

Returns the index of the layer in the coded image to which the coded element belongs.

[VB6]

LayerIndex As Long

read-only

Remarks

If the coded element is a hole, its layer index is defined as that of its parent object.

ECodedElement.LeftLimit

Returns the lowest (integer) X-coordinate of all the pixels of the coded element.

[VB6]

LeftLimit As Long

read-only

Remarks

For a coded element E, this value is defined as: $\lfloor \min \{x \mid (\exists y) (x, y) \in E\} \rfloor$

ECodedElement.MinimumEnclosingRectangle

Returns the Minimum-Area Enclosing Rectangle.

[VB6]

MinimumEnclosingRectangle As ERotatedBoundingBox

read-only

Remarks

The Minimum-Area Enclosing Rectangle is defined as the Feret box with the minimum surface among all the possible orientations.

ECodedElement.MinimumEnclosingRectangleAngle

Returns the angle of the Minimum-Area Enclosing Rectangle.

[VB6]

MinimumEnclosingRectangleAngle As Single

read-only

Remarks

The angle always lies in the range [0 ; pi[.

EcodedElement.MinimumEnclosingRectangleCenter

er

Returns the coordinates of the center of the Minimum-Area Enclosing Rectangle.

[VB6]

MinimumEnclosingRectangleCenter As EPoint

read-only

EcodedElement.MinimumEnclosingRectangleCenterX

X

Returns the abscissa of the center of the Minimum-Area Enclosing Rectangle.

[VB6]

MinimumEnclosingRectangleCenterX As Single

read-only

ECodedElement.MinimumEnclosingRectangleCenterY

Returns the ordinate of the center of the Minimum-Area Enclosing Rectangle.

[VB6]

MinimumEnclosingRectangleCenterY As Single

read-only

ECodedElement.MinimumEnclosingRectangleHeight

Returns the height of the Minimum-Area Enclosing Rectangle.

[VB6]

MinimumEnclosingRectangleHeight As Single

read-only

ECodedElement.MinimumEnclosingRectangleWidth

Returns the width of the Minimum-Area Enclosing Rectangle.

[VB6]

MinimumEnclosingRectangleWidth As Single

read-only

EcodedElement.RenderMask

Creates a Flexible Mask from the coded element.

[VB6]

```
void RenderMask(
    EROIBW8 destination,
    Long offsetX,
    Long offsetY
)
void RenderMask(
    EROIBW8 destination
)
```

Parameters

destination

The image in which the generated mask will be stored.

offsetX

The X-offset that must be applied to bring the zero X-coordinate in the coded image on the first column of the result image (defaults to zero).

offsetY

The Y-offset that must be applied to bring the zero Y-coordinate in the coded image on the first row of the result image (defaults to zero).

Remarks

The size of the result image will not be changed: It must be properly sized beforehand.

EcodedElement.RightLimit

Returns the highest (integer) X-coordinate of all the pixels of the coded element.

[VB6]

RightLimit As Long

read-only

Remarks

For a coded element E, this value is defined as:
$$\text{RightLimit} = \max\{x \mid (\exists y) (x, y) \in E\}$$

EcodedElement.RunCount

Returns the number of runs inside the coded element.

[VB6]

RunCount As Long

read-only

EcodedElement.RunsIterator

Returns an iterator to the runs of the coded element.

[VB6]

RunsIterator As EObjectRunsIterator

read-only

EcodedElement.SigmaX

Returns the centered moment of inertia around X (average squared X-deviation).

[VB6]

SigmaX As Single

read-only

ECodedElement.SigmaXX

Returns the centered cross moment of inertia (average X-deviation * Y-deviation).

[VB6]

SigmaXX As Single

read-only

ECodedElement.SigmaXY

Returns the reduced, centered moment of inertia (around the principal inertia axis).

[VB6]

SigmaXY As Single

read-only

ECodedElement.SigmaY

Returns the centered moment of inertia around Y (average squared Y-deviation).

[VB6]

SigmaY As Single

read-only

ECodedElement.SigmaYY

Returns the reduced, centered moment of inertia (around the secondary inertia axis).

[VB6]

SigmaYY As Single

read-only

ECodedElement.TopLimit

Returns the lowest (integer) Y-coordinate of all the pixels of the coded element.

[VB6]

TopLimit As Long

read-only

Remarks

For a coded element E, this value is defined as:
$$\lfloor \min \{y \mid (\exists x) (x, y) \in E\} \rfloor$$

EcodedImage Class

This class handles runs, objects and features in EasyObject.

Remarks

These entities are stored into three separate dynamic lists for efficient storage. This class pertains to the EasyObject legacy API and should not be used for new developments. It has been replaced by [EcodedImage2](#).

PROPERTIES

BlackClass

Black class index (below the lower threshold).

Connexity

Connexity mode, that is how neighboring pixels are considered to belong to the same objects.

Continuous

Flag indicating whether the objects have to be built in the continuous mode or in the normal mode.

CurrentObjPtr

Pointer to the current objects list item, or **NULL**.

CurrentRunPtr

Pointer to the current run list item, or **NULL**.

DrawDiagonals

Flag indicating whether the limit rectangle diagonals must be drawn or not.

FirstObjPtr	Pointer to the first objects list item, or NULL .
HighColorThreshold	Upper threshold (color) used for image segmentation.
HighImage	Image used as an adaptive upper threshold.
HighThreshold	Upper threshold (gray level) used for image segmentation.
LastObjPtr	Pointer to the last objects list item, or NULL .
LimitAngle	Angle of the skewed bounding box feature, in the current angle unit.
LowColorThreshold	Lower threshold (color) used for image segmentation.
LowImage	Image used as an adaptive lower threshold.
LowThreshold	Lower threshold (gray level) used for image segmentation.
MaxObjects	Maximum number of objects to look for.
NeutralClass	Neutral class index (between both thresholds).

NumFeatures	Number of features currently in use.
NumHoleRuns	Total number of hole runs in the list of object runs.
NumObjects	Number of objects in the coded image.
NumRuns	Total number of runs in the list of object runs.
NumSelectedObjects	Number of objects currently selected.
Threshold	Threshold mode (gray level) used for image segmentation.
ThresholdImage	Single threshold used for image segmentation.
TrueThreshold	Absolute threshold level, when using a single threshold.
WhiteClass	M White class index (above the upper threshold).
THODS	
AddFeat	Adds a feature to the list of features.
AddRunToObj	-

AnalyseObjects	After an image segmentation (see ECodedImage::BuildObjects), computes the values of given "features", i.e. geometric parameters.
BlankFeatures	Resets all values of all features.
BuildHoles	Creates holes.
BuildLabeledObjects	Segments an image into connected blobs comprised of pixels of the same class.
BuildLabeledRuns	Extracts the runs from the image by comparing adjacent pixel values.
BuildObjects	Groups runs to form separated objects (connected blobs), from runs detected by ECodedImage::BuildRuns or from a source image/ROI.
BuildRuns	Converts the specified ROI to classes, and extracts the runs from it.
DetachRunsFromObj	-
DrawObject	Draws the designated object in solid color.
DrawObjectFeature	Draws a graphical representation of a feature of the designated object in solid color.

DrawObjectFeatureWithCurrentPen	Draws a graphical representation of a feature of the designated object in solid color.
DrawObjects	Draws all objects in solid color.
DrawObjectsFeature	Draws a graphical representation of a feature.
DrawObjectsFeatureWithCurrentPen	Draws a graphical representation of a feature.
DrawObjectsWithCurrentPen	Draws all objects in solid color.

DrawObjectWithCurrentPen	Draws the designated object in solid color.
ECodedImage	Constructs a void coded image.
FeatureAverage	Computes the average of the features of all currently selected objects.
FeatureDeviation	Computes the average and standard deviation of the features of all currently selected objects.
FeatureMaximum	Computes the maximum of the features of all currently selected objects.
FeatureMinimum	Computes the minimum of the features of all currently selected objects.
FeatureVariance	Computes the average and variance of the features of all currently selected objects.
GetCurrentObjData	Returns the data of the current object.
GetCurrentRunData	Returns the data of the current run.
GetFeatData	Gets the EFeatureData associated to a given feature list item.

GetFeatDataSize	Returns the data size of the specified feature.
GetFeatDataType	Returns the data type of the specified feature.
GetFeatNum	Returns the code number of the specified feature.
GetFeatPtrByNum	Returns a pointer to the feature list item for a given feature number, or NULL .
GetFeatSize	Returns the size (number of elements) of the feature array for the specified feature.
GetFirstHole	Returns a pointer to the first hole related to the specified object.
GetFirstObjData	Moves the cursor to the first object and returns the associated data.
GetFirstRunData	Moves the cursor to the first run and returns the associated data.
GetFirstRunPtr	Pointer to the first run list item, or NULL .
GetHoleParentObject	Returns a pointer to the real object including the specified hole.

GetLastObjData	Moves the cursor to the last object and returns the associated data.
GetLastRunData	Moves the cursor to the last run and returns the associated data.
GetLastRunPtr	Pointer to the last run list item, or NULL .
GetNextHole	Returns a pointer to the hole following the specified hole, in the objects list.
GetNextObjData	Moves the cursor to the next object and returns the associated data.
GetNextObjPtr	Returns a pointer to the next objects list item, or NULL .
GetNextRunData	Moves the cursor to the next run and returns the associated data.
GetNextRunPtr	Returns a pointer to the next run list item, or NULL .
GetNumHoles	Returns the number of holes related to the specified object.

GetNumObjectRuns	Returns the number of runs comprised in a given object.
GetObjDataPtr	Gets the EObjectData associated to a given objects list item.
GetObjectData	If an object identification number is given, moves the cursor to the object with a given identification number and returns the associated data. If a list item is given, returns the object data associated with an object list item (cf. EListItem). See also ECodedImage::GetCurrentObjData .
GetObjectFeature	Allows retrieving the value of a feature of a given object.
GetObjFirstRunPtr	Returns a pointer to the first run list item of an object.
GetObjLastRunPtr	Returns a pointer to the last run list item of an object.
GetObjPtr	Returns a pointer to the given objects list item.
GetObjPtrByCoordinates	Returns a pointer to the objects list item that contains the point of given coordinates, or NULL .
GetObjPtrByPos	Returns a pointer to the objects list item of given absolute position, or NULL .

GetPreviousObjData	Moves the cursor to the previous object and returns the associated data.
GetPreviousObjPtr	Returns a pointer to the previous objects list item, or NULL .
GetPreviousRunData	Moves the cursor to the previous run and returns the associated data.
GetPreviousRunPtr	Returns a pointer to the previous run list item, or NULL .
GetRunData	Returns the run data associated to the specified run.
GetRunDataPtr	Gets the ERunData associated to a given run list item.
GetRunPtr	Returns a pointer to the run list item of given absolute position, or NULL .
GetRunPtrByCoordinates	Returns a pointer to the run list item that contains the point of given coordinates, or NULL .
IsHole	Returns TRUE if the specified object is a hole, FALSE otherwise.
IsObjectSelected	Sets bSelected to TRUE if an object is selected.

ObjectConvexHull	Computes the convex hull of an object and stores it in a EPathVector .
RemoveAllFeats	Deletes all features from the features list.
RemoveAllObjects	Deletes all objects from the objects list.
RemoveAllRuns	Deletes all runs from the runs list.
RemoveHoles	Permanently erases, from the objects list, holes related to the specified object.
RemoveObject	Deletes an object from the objects list.
RemoveRun	Deletes a run from the runs list.
ResetContinuousMode	When the continuous mode is activated, this method resets the sequence of images.
SelectAllObjects	Selects all objects.
SelectHoles	Selects the holes related to the specified object.
SelectObject	Selects an object.

[SelectObjectsUsingFeature](#)

Selects or deselects objects, according to the value of a specified feature.

[SelectObjectsUsingPosition](#)

Selects or deselects objects, using a delimiting rectangle.

[SetFeatInfo](#)

Sets the appropriate data size and type for a predefined feature.

[SetFirstRunPtr](#)

Sets the first run list item of an object.

[SetLastRunPtr](#)

Sets the last run list item of an object.

[SortObjectsUsingFeature](#)

Sorts objects according to the value of some feature.

[UnselectAllObjects](#)

Deselects all objects.

[UnselectHoles](#)

Unselects the holes related to the specified object.

[UnselectObject](#)

Deselects an object.

ECodedImage.AddFeat

Adds a feature to the list of features.

[VB6]

```
void AddFeat(  
    EFeatureData feature,  
    Long numberOfObjects  
)
```

Parameters

feature

Pointer to an [EFeatureData](#) describing the feature.

numberOfObjects

Number of objects for which the feature will be stored.

ECodedImage.AddRunToObj

-

[VB6]

```
void AddRunToObj(  
    EListItem pObjectToAddTo  
)
```

Parameters

pObjectToAddTo

-

ECodedImage.AnalyseObjects

After an image segmentation (see [ECodedImage::BuildObjects](#)), computes the values of given "features", i.e. geometric parameters.

[VB6]

```
void AnalyseObjects(
    ELegacyFeature feature1,
    ELegacyFeature feature2,
    ELegacyFeature feature3,
    ELegacyFeature feature4,
    ELegacyFeature feature5,
    ELegacyFeature feature6,
    ELegacyFeature feature7,
    ELegacyFeature feature8,
    ELegacyFeature feature9,
    ELegacyFeature feature10
)
```

Parameters

feature1

 Feature code, as defined by [ELegacyFeature](#).

feature2

 Feature code, as defined by [ELegacyFeature](#).

feature3

 Feature code, as defined by [ELegacyFeature](#).

feature4

 Feature code, as defined by [ELegacyFeature](#).

feature5

 Feature code, as defined by [ELegacyFeature](#).

feature6

 Feature code, as defined by [ELegacyFeature](#).

feature7

 Feature code, as defined by [ELegacyFeature](#).

feature8

 Feature code, as defined by [ELegacyFeature](#).

feature9

Feature code, as defined by [ELegacyFeature](#).

feature10

Feature code, as defined by [ELegacyFeature](#).

ECodedImage.BlackClass

Black class index (below the lower threshold).

[VB6]

BlackClass As Integer

read-write

Remarks

Non zero when the black runs (below the lower threshold) are coded. **0** means "do not code this class". <!-- **1** by default. -->

ECodedImage.BankFeatures

Resets all values of all features.

[VB6]

```
void BlankFeatures(  
    )
```

ECodedImage.BuildHoles

Creates holes.

```
[VB6]
void BuildHoles(
    )
void BuildHoles(
    EListItem object_
    )
```

Parameters

object_

Pointer to the objects list item, for which the holes have to be computed.

Remarks

If no argument, the holes are related to all the previously selected real objects. If holes already exist (resulting from a previous call to the [ECodeImage::BuildHoles](#) function), they will be removed from the objects list before the new hole building. Otherwise, the holes are related only to the specified object. Previously created holes are not removed before the new holes are built. If holes related to **object** have already been constructed, they won't be recreated. If **object** is a hole or is **NULL**, no hole will be built. The newly created holes will be added to the list of the objects found in the image. Building holes requires two preliminary steps: the construction of real objects and the selection of objects on which the hole detection has to be performed. At the end of the object construction, all the objects are selected.

ECodeImage.BuildLabeledObjects

Segments an image into connected blobs comprised of pixels of the same class.

```
[VB6]
void BuildLabeledObjects(
    EROIBW8 sourceImage
    )
void BuildLabeledObjects(
    EROIBW16 sourceImage
    )
```

Parameters

sourceImage

Pointer to a source ROI.

Remarks

Uses [EBW8 \(EBW16\)](#) information for class indices, i.e. 255 (65,535) possible classes. Class 0 is not coded. Building objects is the process of grouping pixels from an image to form connected blobs. The pixels are assigned class indices based either on thresholding ([ECodedImage::BuildObjects](#)) or on the pixel values themselves ([BuildLabeledObjects](#)). A blob is a set of connected pixels of the same class.

ECodedImage.BuildLabeledRuns

Extracts the runs from the image by comparing adjacent pixel values.

```
[VB6]
void BuildLabeledRuns(
    EROIBW8 sourceImage
)
void BuildLabeledRuns(
    EROIBW16 sourceImage
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

Remarks

Uses [EBW8 \(EBW16\)](#) information for class indices, i.e. 255 (65,535) possible classes. Class 0 is not coded. Building runs is the process of grouping pixels from an image to form horizontal segments. The pixels are assigned class indices based either on thresholding ([ECodedImage::BuildRuns](#)) or on the pixel values themselves ([BuildLabeledRuns](#)). A run is a set of horizontally connected pixels of the same class.

ECodedImage.BuildObjects

Groups runs to form separated objects (connected blobs), from runs detected by [ECodedImage::BuildRuns](#) or from a source image/ROI.

```
[VB6]

void BuildObjects(
    EROIBW1 sourceImage
)

void BuildObjects(
    EROIBW8 sourceImage
)

void BuildObjects(
    EROIC24 sourceImage
)

void BuildObjects(
    )
```

Parameters

sourceImage

Pointer to a source ROI.

Remarks

Without argument, the method groups the runs detected by [ECodedImage::BuildRuns](#) to form separate objects, i.e. connected components. With a source ROI as argument, the method segments it into connected blobs comprised of pixels of the same class. The [EROIBW8](#) parameter is converted to white/neutral/black classes, using two thresholds. The [EROIC24](#) parameter is converted to white/black classes, using two color thresholds. Then, the method extracts runs from them, and groups the runs to form separate objects, i.e. connected components. Building objects is the process of grouping pixels from an image to form connected blobs. The pixels are assigned class indices based either on thresholding ([BuildObjects](#)) or on the pixel values themselves ([ECodedImage::BuildLabeledObjects](#)). A blob is a set of connected pixels of the same class.

ECodedImage.BuildRuns

Converts the specified ROI to classes, and extracts the runs from it.

```
[VB6]
```

```

void BuildRuns(
    EROIBW1 sourceImage
)
void BuildRuns(
    EROIBW8 sourceImage
)
void BuildRuns(
    EROIC24 sourceImage
)

```

Parameters

sourceImage

Pointer to the source ROI.

Remarks

The **EROIBW8** parameter is converted to white/neutral/black classes, using two thresholds. The **EROIC24** parameter is converted to white/black classes, using two color thresholds. Then, the method extracts runs from them. Building runs is the process of grouping pixels from an image to form horizontal segments. The pixels are assigned class indices based either on thresholding (**BuildRuns**) or on the pixel values themselves ([ECodeImage::BuildLabeledRuns](#)). A run is a set of horizontally connected pixels of the same class.

ECodeImage.Connexity

Connexity mode, that is how neighboring pixels are considered to belong to the same objects.

[VB6]

Connexity As EConnexity

read-write

ECodedImage.Continuous

Flag indicating whether the objects have to be built in the continuous mode or in the normal mode.

[VB6]

Continuous As Boolean

read-write

Remarks

TRUE if objects are built in the continuous mode, **FALSE** if objects are built in the normal mode.

ECodedImage.CurrentObjPtr

Pointer to the current objects list item, or **NULL**.

[VB6]

CurrentObjPtr As EListIItem

read-only

ECodedImage.CurrentRunPtr

Pointer to the current run list item, or **NULL**.

[VB6]

```
CurrentRunPtr As EListIItem
```

read-only

ECodedImage.DetachRunsFromObj

-

[VB6]

```
void DetachRunsFromObj (
    EListIItem pCurrentObject
)
```

Parameters

pCurrentObject

-

ECodedImage.DrawDiagonals

Flag indicating whether the limit rectangle diagonals must be drawn or not.

[VB6]

```
DrawDiagonals As Boolean
```

read-write

Remarks

If **TRUE** (default), diagonals are drawn.

ECodedImage.DrawObject

Draws the designated object in solid color.

[VB6]

```
void DrawObject(
    Long graphicContext,
    Long objectNumber,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawObject(
    Long graphicContext,
    EListIItem object_,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawObject(
    Long graphicContext,
    ERGBCColor color,
    Long objectNumber,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawObject(
    Long graphicContext,
    ERGBCColor color,
    EListIItem object_,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

```

void DrawObject(
    EDrawAdapter graphicContext,
    Long objectNumber,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawObject(
    EDrawAdapter graphicContext,
    EListIItem object_,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

```

Parameters*graphicContext*

Handle of the device context on which to draw.

objectNumber

Number of the object to be drawn.

zoomX

Horizontal zooming factor. By default, true scale is used.

*zoomY*Vertical zooming factor. If set to **0**, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.*panX*

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

object_

Pointer to the list item (from the objects list) corresponding to the object to be drawn.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used. The objects can be specified either by number or by list pointer.

ECodedImage.DrawObjectFeature

Draws a graphical representation of a feature of the designated object in solid color.

[VB6]

```
void DrawObjectFeature(
    Long graphicContext,
    ELegacyFeature feature,
    Long objectNumber,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawObjectFeature(
    Long graphicContext,
    ELegacyFeature feature,
    EListItem object_,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawObjectFeature(
    Long graphicContext,
    ERGBColor color,
    ELegacyFeature feature,
    EListItem object_,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

```

void DrawObjectFeature(
    Long graphicContext,
    ERGBColor color,
    ELegacyFeature feature,
    Long objectNumber,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawObjectFeature(
    EDrawAdapter graphicContext,
    ELegacyFeature feature,
    Long objectNumber,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawObjectFeature(
    EDrawAdapter graphicContext,
    ELegacyFeature feature,
    EListIItem object_,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

```

Parameters

graphicContext

Handle of the device context on which to draw.

feature

Feature to be drawn, as defined by [ELegacyFeature](#) (use any value matching the aforementioned features, f.i. [ELegacyFeature_EllipseWidth](#) will draw the ellipse of inertia).

objectNumber

Number of the object to be drawn.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0**, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

object_

Pointer to the list item (from the objects list) corresponding to the object to be drawn.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used. The objects can be specified either by number or by list pointer. If a required feature has not been computed for some object, nothing is drawn and no error message is issued!

ECodeImage.DrawObjectFeatureWithCurrentPen

Draws a graphical representation of a feature of the designated object in solid color.

[VB6]

```

void DrawObjectFeatureWithCurrentPen(
    Long graphicContext,
    ELegacyFeature feature,
    EListItem objectNumber,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawObjectFeatureWithCurrentPen(
    Long graphicContext,
    ELegacyFeature feature,
    Long objectNumber,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

```

Parameters

graphicContext

Handle of the device context on which to draw.

feature

Feature to be drawn, as defined by [ELegacyFeature](#) (use any value matching the aforementioned features, f.i. [ELegacyFeature_EllipseWidth](#) will draw the ellipse of inertia).

objectNumber

Number of the object to be drawn.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0**, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used. The objects can be specified either by number or by list pointer. If a required feature has not been computed for some object, nothing is drawn and no error message is issued!

ECodeImage.DrawObjects

Draws all objects in solid color.

[VB6]

```
void DrawObjects(
    Long graphicContext,
    ESelectionFlag selectionFlag,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

```
void DrawObjects(
    Long graphicContext,
    ERGBColor color,
    ESelectionFlag selectionFlag,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawObjects(
    EDrawAdapter graphicContext,
    ESelectionFlag selectionFlag,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

selectionFlag

Tells how the selected/unselected state of the objects is handled, as defined by [ESelectionFlag](#).

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used for all objects (to vary the colors, draw the objects separately using the [ECodeImage::DrawObject](#) method instead). Optionally, only the selected or deselected objects can be drawn.

ECodeImage.DrawObjectsFeature

Draws a graphical representation of a feature.

[VB6]

```
void DrawObjectsFeature(
    Long graphicContext,
    ELegacyFeature feature,
    ESelectionFlag selectionFlag,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawObjectsFeature(
    Long graphicContext,
    ERGBColor color,
    ELegacyFeature feature,
    ESelectionFlag selectionFlag,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawObjectsFeature(
    EDrawAdapter graphicContext,
    ELegacyFeature feature,
    ESelectionFlag selectionFlag,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters*graphicContext*

Handle of the device context on which to draw.

*feature*Feature to be drawn, as defined by [ELegacyFeature](#) (use any value matching the aforementioned features, f.i. [ELegacyFeature_EllipseWidth](#) will draw the ellipse of inertia).*selectionFlag*

Tells how the selected/unselected state of the objects is handled, as defined by

[ESelectionFlag](#).*zoomX*

Horizontal zooming factor. By default, true scale is used.

*zoomY*Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.*panX*

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used for all objects (to vary the colors, draw the objects separately using the [ECodedImage::DrawObjectFeature](#) method instead). Optionally, only the selected or deselected objects can be drawn. Only the following features can be drawn: * [ELegacyFeature_GravityCenter](#): upright cross; * [ELegacyFeature_Centroid](#): skewed cross; * [ELegacyFeature_Limit](#): upright bounding rectangle with diagonals; * [ELegacyFeature_Limit45](#): skewed bounding rectangle with diagonals; * [ELegacyFeature_EllipseWidth](#): ellipse of inertia (edge and main axis). If a required feature has not been computed for some object, nothing is drawn and no error message is issued!

ECodedImage.DrawObjectsFeatureWithCurrentPen

Draws a graphical representation of a feature.

[VB6]

```
void DrawObjectsFeatureWithCurrentPen(
    Long graphicContext,
    ELegacyFeature feature,
    ESelectionFlag selectionFlag,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

feature

Feature to be drawn, as defined by [ELegacyFeature](#) (use any value matching the aforementioned features, f.i. [ELegacyFeature_EllipseWidth](#) will draw the ellipse of inertia).

selectionFlag

Tells how the selected/unselected state of the objects is handled, as defined by [ESelectionFlag](#).

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used for all objects (to vary the colors, draw the objects separately using the [ECodedImage::DrawObjectFeature](#) method instead). Optionally, only the selected or deselected objects can be drawn. Only the following features can be drawn: * [ELegacyFeature_GravityCenter](#): upright cross; * [ELegacyFeature_Centroid](#): skewed cross; * [ELegacyFeature_Limit](#): upright bounding rectangle with diagonals; * [ELegacyFeature_Limit45](#): skewed bounding rectangle with diagonals; * [ELegacyFeature_EllipseWidth](#): ellipse of inertia (edge and main axis). If a required feature has not been computed for some object, nothing is drawn and no error message is issued!

ECodedImage.DrawObjectsWithCurrentPen

Draws all objects in solid color.

[VB6]

```
void DrawObjectsWithCurrentPen(
    Long graphicContext,
    ESelectionFlag selectionFlag,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

selectionFlag

Tells how the selected/unselected state of the objects is handled, as defined by [ESelectionFlag](#).

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used for all objects (to vary the colors, draw the objects separately using the [ECodedImage::DrawObject](#) method instead). Optionally, only the selected or deselected objects can be drawn.

ECodedImage.DrawObjectWithCurrentPen

Draws the designated object in solid color.

[VB6]

```
void DrawObjectWithCurrentPen(
    Long graphicContext,
    Long objectNumber,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawObjectWithCurrentPen(
    Long graphicContext,
    EListIItem object_,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

objectNumber

Number of the object to be drawn.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0**, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

object_

Pointer to the list item (from the objects list) corresponding to the object to be drawn.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used. The objects can be specified either by number or by list pointer.

ECodedImage.ECodedImage

Constructs a void coded image.

```
[VB6]  
void ECodedImage(  
    ECodedImage other  
)  
  
void ECodedImage(  
)
```

Parameters

other

-

ECodedImage.FeatureAverage

Computes the average of the features of all currently selected objects.

```
[VB6]  
void FeatureAverage(  
    ELegacyFeature feature,  
    Single average  
)
```

Parameters

feature

Feature code, as defined by [ELegacyFeature](#).

average

Reference to the feature average.

Remarks

This measures the central tendency of a population of objects.

ECodedImage.FeatureDeviation

Computes the average and standard deviation of the features of all currently selected objects.

```
[VB6]  
void FeatureDeviation(  
    ELegacyFeature feature,  
    Single average,  
    Single deviation  
)
```

Parameters

feature
Feature code, as defined by [ELegacyFeature](#).
average
Reference to the feature average.
deviation
Reference to the feature standard deviation.

ECodedImage.FeatureMaximum

Computes the maximum of the features of all currently selected objects.

```
[VB6]
```

```
void FeatureMaximum(
    ELegacyFeature feature,
    Single maximum
)
```

Parameters

feature
Feature code, as defined by [ELegacyFeature](#).
maximum
Reference to the feature maximum.

ECodedImage.FeatureMinimum

Computes the minimum of the features of all currently selected objects.

```
[VB6]
void FeatureMinimum(
    ELegacyFeature feature,
    Single minimum
)
```

Parameters

feature
Feature code, as defined by [ELegacyFeature](#).
minimum
Reference to the feature minimum.

ECodedImage.FeatureVariance

Computes the average and variance of the features of all currently selected objects.

[VB6]

```
void FeatureVariance(  
    ELegacyFeature feature,  
    Single average,  
    Single variance  
)
```

Parameters

feature

Feature code, as defined by [ELegacyFeature](#).

average

Reference to the feature average.

variance

Reference to the feature variance.

Remarks

This measures the central tendency and the dispersion of a population of objects.

ECodedImage.FirstObjPtr

Pointer to the first objects list item, or **NULL**.

[VB6]

```
FirstObjPtr As EListItem
```

read-only

ECodedImage.GetCurrentObjData

Returns the data of the current object.

[VB6]

```
void GetCurrentObjData(  
    EObjectData objectData  
)
```

Parameters

objectData

Pointer to an [EObjectData](#) structure to receive the data.

Remarks

ECodedImage.GetCurrentRunData

Returns the data of the current run.

[VB6]

```
void GetCurrentRunData(  
    ERunData run  
)
```

Parameters

run

Pointer to an [ERunData](#) to receive the data.

ECodedImage.GetFeatData

Gets the [EFeatureData](#) associated to a given feature list item.

[VB6]

```
Boolean GetFeatData(
    EListIItem currentFeature,
    EFeatureData featureData
)
```

Parameters

currentFeature

Pointer to the feature list item.

featureData

Pointer to a [EFeatureData](#) to receive the data.

ECodedImage.GetFeatDataSize

Returns the data size of the specified feature.

[VB6]

```
EDataSize GetFeatDataSize(
    Long position
)

EDataSize GetFeatDataSize(
    EListIItem currentFeature
)
```

Parameters

position

Absolute position in the features list, counting from **0** on.

currentFeature

Pointer to the feature list item.

Remarks

The features data sizes are defined in [EDataSize](#).

ECodedImage.GetFeatDataType

Returns the data type of the specified feature.

```
[VB6]  
EDataType GetFeatDataType(  
    Long position  
)  
  
EDataType GetFeatDataType(  
    EListIItem currentFeature  
)
```

Parameters

position

Absolute position in the features list, counting from **0** on.

currentFeature

Pointer to the feature list item.

Remarks

The features data types are defined in [EDataType](#).

ECodedImage.GetFeatNum

Returns the code number of the specified feature.

```
[VB6]  
Long GetFeatNum(  
    Long position  
)  
  
Long GetFeatNum(  
    EListIItem currentFeature  
)
```

Parameters*position*Absolute position in the features list, counting from **0** on.*currentFeature*

Pointer to the feature list item.

RemarksThe features code numbers are defined in [ELegacyFeature](#).

ECodedImage.GetFeatPtrByNum

Returns a pointer to the feature list item for a given feature number, or **NULL**.

[VB6]

```
EListIItem GetFeatPtrByNum(
    Long numFeat
)
```

Parameters*numFeat*Feature number, as defined by [ELegacyFeature](#).

ECodedImage.GetFeatSize

Returns the size (number of elements) of the feature array for the specified feature.

[VB6]

```
Long GetFeatSize(
    Long position
)

Long GetFeatSize(
    EListIItem currentFeature
)
```

Parameters*position*Absolute position in the features list, counting from **0** on.*currentFeature*

Pointer to the feature list item.

ECodedImage.GetFirstHole

Returns a pointer to the first hole related to the specified object.

[VB6]

```
EListItem GetFirstHole(
    EListItem parentObject
)
```

Parameters*parentObject*

Pointer to the objects list item, for which the first hole has to be pointed out.

Remarks

If **parentObject** refers to a hole (instead of a real object) or to an object comprising no hole, the [ECodedImage::GetFirstHole](#) function returns **NULL**.

ECodedImage.GetFirstObjData

Moves the cursor to the first object and returns the associated data.

[VB6]

```
void GetFirstObjData(
    EObjectData object_
)
```

Parameters

object_

Pointer to an [EObjectData](#) to receive the data.

Remarks

ECodedImage.GetFirstRunData

Moves the cursor to the first run and returns the associated data.

[VB6]

```
void GetFirstRunData(  
    ERunData run  
)
```

Parameters

run

Pointer to an [ERunData](#) to receive the data.

ECodedImage.GetFirstRunPtr

Pointer to the first run list item, or **NULL**.

[VB6]

```
EListItem GetFirstRunPtr(  
)
```

ECodedImage.GetHoleParentObject

Returns a pointer to the real object including the specified hole.

[VB6]

```
EListIItem GetHoleParentObject(  
    EListIItem hole  
)
```

Parameters

hole

Pointer to the hole list item, for which the parent object has to be pointed out.

ECodedImage.GetLastObjData

Moves the cursor to the last object and returns the associated data.

[VB6]

```
void GetLastObjData(  
    EObjectData object_  
)
```

Parameters

object_

Pointer to an [EObjectData](#) structure to receive the data.

ECodedImage.GetLastRunData

Moves the cursor to the last run and returns the associated data.

[VB6]

```
void GetLastRunData(  
    ERunData run  
)
```

Parameters

run

Pointer to an [ERunData](#) to receive the data.

ECodedImage.GetLastRunPtr

Pointer to the last run list item, or **NULL**.

[VB6]

```
EListItem GetLastRunPtr()  
 )
```

ECodedImage.GetNextHole

Returns a pointer to the hole following the specified hole, in the objects list.

[VB6]

```
EListItem GetNextHole(  
 EListItem hole  
 )
```

Parameters

hole

Pointer to the hole list item, for which the following hole has to be pointed out.

Remarks

If there is no hole yet in the list, the [GetNextHole](#) function returns **NULL**.

ECodedImage.GetNextObjData

Moves the cursor to the next object and returns the associated data.

[VB6]

```
void GetNextObjData(
    EObjectData object_
)
```

Parameters

object_

Pointer to an [EObjectData](#) to receive the data.

ECodedImage.GetNextObjPtr

Returns a pointer to the next objects list item, or **NULL**.

[VB6]

```
EListItem GetNextObjPtr(
    EListItem listItem
)
```

Parameters

listItem

Pointer to the current objects list item.

ECodedImage.GetNextRunData

Moves the cursor to the next run and returns the associated data.

[VB6]

```
void GetNextRunData(
    ERunData run
)
```

```
void GetNextRunData(
    ERunData run,
    EListIItem listItem
)
```

Parameters

run

Pointer to an [ERunData](#) to receive the data.

listItem

Pointer to the current run list item.

ECodedImage.GetNextRunPtr

Returns a pointer to the next run list item, or **NULL**.

[VB6]

```
EListIItem GetNextRunPtr(
    EListIItem listItem
)
```

Parameters

listItem

Pointer to the current run list item.

ECodedImage.GetNumHoles

Returns the number of holes related to the specified object.

[VB6]

```
Long GetNumHoles(
    EListIItem object_
)
```

Parameters*object_*

Pointer to the object list item whose holes have to be counted.

Remarks

By default, the parameter **object** is set to **NULL**, meaning that the function returns the total number of holes added to the objects list. After a call to the [EcodedImage::BuildHoles](#) function, the [EcodedImage::NumObjects](#) and [EcodedImage::NumSelectedObjects](#) properties contain the total number of objects (i.e. real objects + holes).

EcodedImage.GetNumObjectRuns

Returns the number of runs comprised in a given object.

[VB6]

```
Long GetNumObjectRuns (
    Long objectNumber
)
Long GetNumObjectRuns (
    EListIItem listItem
)
```

Parameters*objectNumber*

Object identification number.

listItem

Pointer to an objects list item.

ECodedImage.GetObjDataPtr

Gets the [EOBJECTDATA](#) associated to a given objects list item.

```
[VB6]  
Boolean GetObjDataPtr(  
    EListIItem currentFeature,  
    EObjectData objectData  
)
```

Parameters

currentFeature

Pointer to the current objects list item.

objectData

Pointer to a [EOBJECTDATA](#) to receive the data.

ECodedImage.GetObjectData

If an object identification number is given, moves the cursor to the object with a given identification number and returns the associated data. If a list item is given, returns the object data associated with an object list item (cf. [EListIItem](#)). See also [ECodedImage::GetCurrentObjData](#).

```
[VB6]  
void GetObjectData(  
    EObjectData object_,  
    Long objectNumber  
)  
  
void GetObjectData(  
    EObjectData object_,  
    EListIItem listItem  
)
```

Parameters

object_

Pointer to an [EObjectData](#) structure to receive the object data.

objectNumber

Object identification number.

listItem

Pointer to the current object list item (cf. [EListItem](#)).

EEncodedImage.GetObjectFeature

Allows retrieving the value of a feature of a given object.

[VB6]

```
void GetObjectFeature(
    ELegacyFeature feature,
    EListItem objectNumber,
    Unsupported variant type result
)

void GetObjectFeature(
    ELegacyFeature feature,
    EListItem object_,
    Integer result
)

void GetObjectFeature(
    ELegacyFeature feature,
    EListItem object_,
    Long result
)

void GetObjectFeature(
    ELegacyFeature feature,
    EListItem object_,
    Single result
)

void GetObjectFeature(
    ELegacyFeature feature,
    EListItem object_,
    Double result
)
```

```
void GetObjectFeature(
    Long feature,
    Long objectNumber,
    Unsupported variant type result
)

void GetObjectFeature(
    Long feature,
    Long objectNumber,
    Integer result
)

void GetObjectFeature(
    Long feature,
    Long objectNumber,
    Long result
)

void GetObjectFeature(
    Long feature,
    Long objectNumber,
    Single result
)

void GetObjectFeature(
    Long feature,
    Long objectNumber,
    Double result
)

void GetObjectFeature(
    EListItem feature,
    Long objectNumber,
    Unsupported variant type result
)

void GetObjectFeature(
    EListItem feature,
    Long objectNumber,
    Integer result
)

void GetObjectFeature(
    EListItem feature,
    Long objectNumber,
    Long result
)
```

```

void GetObjectFeature(
    EListItem feature,
    Long objectNumber,
    Single result
)

void GetObjectFeature(
    EListItem feature,
    Long objectNumber,
    Double result
)

```

Parameters*feature*

Pointer to the feature list item.

objectNumber

Object number.

result

Reference to the feature value.

object_

Pointer to the list item (from the objects list) corresponding to the object.

EEncodedImage.GetObjFirstRunPtr

Returns a pointer to the first run list item of an object.

[VB6]

```

EListItem GetObjFirstRunPtr(
    Long objectNumber
)

EListItem GetObjFirstRunPtr(
    EListItem listItem
)

```

Parameters*objectNumber*

Object identification number.

listItem

Pointer to the objects list item.

ECodedImage.GetObjLastRunPtr

Returns a pointer to the last run list item of an object.

```
[VB6]  
EListItem GetObjLastRunPtr(  
    Long objectNumber  
)  
  
EListItem GetObjLastRunPtr(  
    EListItem objectNumber  
)
```

Parameters

objectNumber

Object identification number.

ECodedImage.GetObjPtr

Returns a pointer to the given objects list item.

```
[VB6]  
EListItem GetObjPtr(  
    Long objectNumber  
)
```

Parameters

objectNumber

Object identification number.

ECodedImage.GetObjPtrByCoordinates

Returns a pointer to the objects list item that contains the point of given coordinates, or **NULL**.

[VB6]

```
EListItem GetObjPtrByCoordinates(
    Long x,
    Long y
)
```

Parameters

x

Point abscissa.

y

Point ordinate.

Remarks

This function is useful for object selection with a mouse.

ECodedImage.GetObjPtrByPos

Returns a pointer to the objects list item of given absolute position, or **NULL**.

[VB6]

```
EListItem GetObjPtrByPos(
    Long position
)
```

Parameters

position

Absolute position in the objects list, counting from **0** on.

ECodedImage.GetPreviousObjData

Moves the cursor to the previous object and returns the associated data.

```
[VB6]  
void GetPreviousObjData(  
    EObjectData object_  
)
```

Parameters

object_
Pointer to an [EObjectData](#) to receive the data.

ECodedImage.GetPreviousObjPtr

Returns a pointer to the previous objects list item, or **NULL**.

```
[VB6]  
EListItem GetPreviousObjPtr(  
    EListItem listItem  
)
```

Parameters

listItem
Pointer to the current objects list item.

ECodedImage.GetPreviousRunData

Moves the cursor to the previous run and returns the associated data.

```
[VB6]

void GetPreviousRunData(
    ERunData run,
    EListItem listItem
)

void GetPreviousRunData(
    ERunData run
)
```

Parameters

run
Pointer to an [ERunData](#) to receive the data.
listItem
Pointer to the current run list item.

ECodedImage.GetPreviousRunPtr

Returns a pointer to the previous run list item, or **NULL**.

```
[VB6]

EListItem GetPreviousRunPtr(
    EListItem listItem
)
```

Parameters

listItem
Pointer to the current run list item.

ECodedImage.GetRunData

Returns the run data associated to the specified run.

```
[VB6]

void GetRunData(
    ERunData run,
    Long position
)

void GetRunData(
    ERunData run,
    EListItem listItem
)
```

Parameters

run

Pointer to an [ERunData](#) to receive the data.

position

Absolute position in the run list, counting from **0** on.

listItem

Pointer to the current run list item.

ECodedImage.GetRunDataPtr

Gets the [ERunData](#) associated to a given run list item.

```
[VB6]
```

```
Boolean GetRunDataPtr(
    EListItem currentFeature,
    ERunData runData
)
```

Parameters

currentFeature

Pointer to the current run list item.

runData

Pointer to a [ERunData](#) to receive the data.

ECodedImage.GetRunPtr

Returns a pointer to the run list item of given absolute position, or **NULL**.

```
[VB6]  
EListItem GetRunPtr(  
    Long position  
)
```

Parameters

position

Absolute position in the run list, counting from 0 on.

ECodedImage.GetRunPtrByCoordinates

Returns a pointer to the run list item that contains the point of given coordinates, or **NULL**.

```
[VB6]  
EListItem GetRunPtrByCoordinates(  
    Long x,  
    Long y  
)
```

Parameters

x

Point abscissa.

y

Point ordinate.

Remarks

This function is useful for run selection with a mouse.

ECodedImage.HighColorThreshold

Upper threshold (color) used for image segmentation.

[VB6]

HighColorThreshold As EC24

read-write

Remarks

The threshold value is constant over the whole image.

ECodedImage.HighImage

Image used as an adaptive upper threshold.

[VB6]

HighImage As EROIBW8

read-write

Remarks

The threshold is adaptive (specified pixel by pixel).

ECodedImage.HighThreshold

Upper threshold (gray level) used for image segmentation.

[VB6]

HighThreshold As Long

read-write

Remarks

The threshold value is constant over the whole image.

ECodedImage.IsHole

Returns **TRUE** if the specified object is a hole, **FALSE** otherwise.

[VB6]

```
Boolean IsHole(  
    EListIItem object_  
)
```

Parameters

object_

Pointer to the objects list item.

ECodedImage.IsObjectSelected

Sets **bSelected** to **TRUE** if an object is selected.

[VB6]

```
void IsObjectSelected(  
    Long objectNumber,  
    Boolean selected  
)
```

```
void IsObjectSelected(
    EListIItem listItem,
    Boolean selected
)
```

Parameters

objectNumber

Object identification number.

selected

Reference to the result.

listItem

Pointer to the objects list item.

Remarks

ECodedImage.LastObjPtr

Pointer to the last objects list item, or **NULL**.

[VB6]

LastObjPtr As EListIItem

read-only

ECodedImage.LimitAngle

Angle of the skewed bounding box feature, in the current angle unit.

[VB6]

LimitAngle As Single

read-write

ECodedImage.LowColorThreshold

Lower threshold (color) used for image segmentation.

[VB6]

LowColorThreshold As EC24

read-write

Remarks

The threshold value is constant over the whole image.

ECodedImage.LowImage

Image used as an adaptive lower threshold.

[VB6]

LowImage As EROIBW8

read-write

Remarks

The threshold value is adaptive (specified pixel by pixel).

ECodedImage.LowThreshold

Lower threshold (gray level) used for image segmentation.

[VB6]

LowThreshold As Long

read-write

Remarks

The threshold value is constant over the whole image.

ECodedImage.MaxObjects

Maximum number of objects to look for.

[VB6]

MaxObjects As Long

read-write

Remarks

After having found that amount of object, the process will stop and return an error message.
If not set, no maximum value is defined.

ECodedImage.NeutralClass

Neutral class index (between both thresholds).

[VB6]

NeutralClass As Integer

read-write

Remarks

Non zero when the neutral runs (between both thresholds) are coded. **0** means "do not code this class". <!-- **2** by default. -->

ECodedImage.NumFeatures

Number of features currently in use.

[VB6]

NumFeatures As Long

read-only

ECodedImage.NumHoleRuns

Total number of hole runs in the list of object runs.

[VB6]

NumHoleRuns As Long

read-only

Remarks

After a call to [ECodedImage::BuildHoles](#), [ECodedImage::NumRuns](#) contains the total number of runs (real object runs + hole runs).

ECodedImage.NumObjects

Number of objects in the coded image.

[VB6]

NumObjects As Long

read-only

Remarks

After a call to [ECodedImage::BuildHoles](#), [ECodedImage::NumObjects](#) returns the total number of objects (real objects + holes).

ECodedImage.NumRuns

Total number of runs in the list of object runs.

[VB6]

NumRuns As Long

read-only

Remarks

After a call to [ECodedImage::BuildHoles](#), [ECodedImage::NumRuns](#) returns the total number of runs (real object runs + hole runs).

ECodedImage.NumSelectedObjects

Number of objects currently selected.

[VB6]

NumSelectedObjects As Long

read-write

Remarks

After a call to [ECodedImage::BuildHoles](#), [ECodedImage::NumSelectedObjects](#) returns the total number of objects (real objects + holes).

ECodedImage.ObjectConvexHull

Computes the convex hull of an object and stores it in a [EPathVector](#).

[VB6]

```
void ObjectConvexHull(
    EListIItem object_,
    EPathVector destinationVector
)
```

Parameters

object_

Pointer to the objects list item.

destinationVector

Vector containing the vertices coordinates of the convex hull.

ECodedImage.RemoveAllFeats

Deletes all features from the features list.

[VB6]

```
void RemoveAllFeats(
)
```

ECodedImage.RemoveAllObjects

Deletes all objects from the objects list.

[VB6]

```
void RemoveAllObjects(  
)
```

ECodedImage.RemoveAllRuns

Deletes all runs from the runs list.

[VB6]

```
void RemoveAllRuns(  
)
```

ECodedImage.RemoveHoles

Permanently erases, from the objects list, holes related to the specified object.

[VB6]

```
void RemoveHoles(  
EListIItem object_  
)
```

Parameters

object_

Pointer to the objects list item, for which the holes have to be erased.

Remarks

If the parameter is **NULL**, all the holes are deleted. By default, the parameter is set to **NULL**, meaning that all the holes have to be erased from the objects list.

ECodedImage.RemoveObject

Deletes an object from the objects list.

```
[VB6]  
void RemoveObject(  
    Long objectNumber  
)  
  
void RemoveObject(  
    EListItem listItem  
)
```

Parameters

objectNumber

Object identification number.

listItem

Pointer to the current objects list item.

ECodedImage.RemoveRun

Deletes a run from the runs list.

```
[VB6]  
void RemoveRun(  
    Long position  
)  
  
void RemoveRun(  
    EListItem listItem  
)
```

Parameters

position

Absolute position in the run list, counting from 0 on.

listItem

Pointer to the current run list item.

ECodedImage.ResetContinuousMode

When the continuous mode is activated, this method resets the sequence of images.

[VB6]

```
void ResetContinuousMode()  
 )
```

Remarks

Thus, the next call for an object building will not take into account any previous image. If the continuous mode is disabled, this method does nothing.

ECodedImage.SelectAllObjects

Selects all objects.

[VB6]

```
void SelectAllObjects()  
 )
```

ECodedImage.SelectHoles

Selects the holes related to the specified object.

[VB6]

```
void SelectHoles(
    EListIItem parentObject
)
```

Parameters

parentObject

Pointer to the objects list item, for which the holes have to be selected.

Remarks

If the parameter is **NULL**, all the holes are selected. By default, the parameter is set to **NULL**, meaning that all the holes have to be selected. If **parentObject** is a hole (instead of a real object) or has no hole, no selection is performed.

ECodedImage.SelectObject

Selects an object.

[VB6]

```
void SelectObject(
    Long objectNumber
)

void SelectObject(
    EListIItem listItem
)
```

Parameters

objectNumber

Object identification number.

listItem

Pointer to the objects list item.

ECodedImage.SelectObjectsUsingFeature

Selects or deselects objects, according to the value of a specified feature.

[VB6]

```
void SelectObjectsUsingFeature(
    ELegacyFeature feature,
    Unsupported variant type minimum,
    Unsupported variant type maximum,
    ESelectOption operation
)

void SelectObjectsUsingFeature(
    ELegacyFeature feature,
    Integer minimum,
    Integer maximum,
    ESelectOption operation
)

void SelectObjectsUsingFeature(
    ELegacyFeature feature,
    Long minimum,
    Long maximum,
    ESelectOption operation
)

void SelectObjectsUsingFeature(
    ELegacyFeature feature,
    Single minimum,
    Single maximum,
    ESelectOption operation
)

void SelectObjectsUsingFeature(
    ELegacyFeature feature,
    Double minimum,
    Double maximum,
    ESelectOption operation
)
```

Parameters

feature

Feature code, as defined by [EFeature](#).

minimum

Selection interval lower bound.

maximum

Selection interval upper bound.

operation

Selection mode, as defined by [ESelectOption](#).

ECodedImage.SelectObjectsUsingPosition

Selects or deselects objects, using a delimiting rectangle.

[VB6]

```
void SelectObjectsUsingPosition(
    EBaseROI roi,
    ESelectByPosition operation
)

void SelectObjectsUsingPosition(
    Long originX,
    Long originY,
    Long width,
    Long height,
    ESelectByPosition operation
)
```

Parameters

roi

Pointer to an image/ROI whose position parameters will define the selection rectangle.

operation

Selection mode, as defined by [ESelectByPosition](#).

originX

Abscissa of the upper left corner of the rectangle.

originY

Ordinate of the upper left corner of the rectangle.

width

Rectangle width, in pixels.

height

Rectangle height, in pixels.

Remarks

The rectangle coordinates are always specified with respect to the whole image.

EcodedImage.SetFeatInfo

Sets the appropriate data size and type for a predefined feature.

```
[VB6]  
void SetFeatInfo(  
    EFeatureData feature,  
    ELegacyFeature featureCode  
)
```

Parameters

feature

Feature code, as defined by [ELegacyFeature](#).

featureCode

Pointer to a [EFeatureData](#) structure describing the feature.

EcodedImage.SetFirstRunPtr

Sets the first run list item of an object.

```
[VB6]  
void SetFirstRunPtr(  
    EListItem firstRun,  
    Long objectNumber  
)  
  
void SetFirstRunPtr(  
    EListItem firstRun,  
    EListItem currentObject  
)
```

Parameters

firstRun

Pointer to the first run of the object.

objectNumber

Object identification number.

currentObject

Pointer to the objects list item.

ECodedImage.SetLastRunPtr

Sets the last run list item of an object.

[VB6]

```
void SetLastRunPtr(
    EListItem lastRun,
    Long objectNumber
)

void SetLastRunPtr(
    EListItem lastRun,
    EListItem currentObject
)
```

Parameters

lastRun

Pointer to the last run of the object.

objectNumber

Object identification number.

currentObject

Pointer to the objects list item.

ECodedImage.SortObjectsUsingFeature

Sorts objects according to the value of some feature.

[VB6]

```
void SortObjectsUsingFeature(
    ELegacyFeature feature,
    ESortOption Operation
)
```

Parameters

feature

Feature code, as defined by [ELegacyFeature](#).

Operation

Selection mode, as defined by [ESortOption](#).

ECodedImage.Threshold

Threshold mode (gray level) used for image segmentation.

[VB6]

Threshold As Long

read-write

Remarks

By default, the "minimum residue" mode is used to determine the threshold. The threshold is constant over the whole image. When using a single threshold instead of a low threshold and a high threshold, the neutral class is ignored. Only the black and white classes are relevant.

ECodedImage.ThresholdImage

Single threshold used for image segmentation.

[VB6]

ThresholdImage As EROIBW8

read-write

Remarks

The threshold value is adaptive (specified pixel by pixel). When using a single threshold instead of a low threshold and a high threshold, the neutral class is ignored. Only the black and white classes are relevant.

ECodedImage.TrueThreshold

Absolute threshold level, when using a single threshold.

[VB6]

TrueThreshold As Long

read-only

ECodedImage.UnselectAllObjects

Deselects all objects.

[VB6]

```
void UnselectAllObjects()  
 )
```

ECodedImage.UnselectHoles

Unselects the holes related to the specified object.

[VB6]

```
void UnselectHoles(
    EListIItem parentObject
)
```

Parameters*parentObject*

Pointer to the objects list item, for which the holes have to be unselected.

Remarks

If the parameter is **NULL**, all the holes are unselected. By default, the parameter is set to **NULL**, meaning that all the holes have to be unselected. If **parentObject** is a hole (instead of a real object) or if **parentObject** has no hole, nothing is changed.

ECodedImage.UnselectObject

Deselects an object.

[VB6]

```
void UnselectObject(
    Long objectNumber
)

void UnselectObject(
    EListIItem listItem
)
```

Parameters*objectNumber*

Object identification number.

listItem

Pointer to the objects list item.

Remarks

Once an object has been unselected, it doesn't allow browsing a list of selected objects anymore (using [ECodedImage::GetPreviousObjPtr](#) or [ECodedImage::GetNextObjPtr](#)).

ECodedImage.WhiteClass

White class index (above the upper threshold).

[VB6]

WhiteClass As Integer

read-write

Remarks

Non zero when the white runs (above the upper threshold) are coded. **0** means "do not code this class". <!-- **3** by default. -->

ECodedImage2 Class

The main class of the EasyObject API that represents a coded image, as produced by the encoder.

Remarks

It provides methods to get features of the encoded image, to access an object in a particular layer of the encoded image, to draw objects or objects features in a particular layer of the encoded image, to render a mask, etc...

PROPERTIES

Height

Returns the height of the coded image.

LayerCount

Returns the number of layers that are encoded.

[StartY](#)

Returns the lowest row index, for all the runs of all the objects in the coded image.

[Width](#)

M Returns the width of the coded image.

E

THODS

[ClearFeatureCache](#)

Clears the internal cache for the computed features.

[Draw](#)

Draw the designated coded element, all the objects from a layer, or all the coded elements from a given selection.

[DrawFeature](#)

Draw a given feature of the designated coded element, of all the objects from a layer, or of all the coded elements from a given selection.

[DrawFeatureWithCurrentPen](#)

Draw a given feature of the designated coded element, of all the objects from a layer, or of all the coded elements from a given selection.

[DrawHole](#)

Draw the designated hole.

[DrawHoleFeature](#)

Draw a given feature of the designated hole.

DrawHoleFeatureWithCurrentPen	Draw a given feature of the designated hole.
DrawHoleWithCurrentPen	-
DrawObject	Draw the designated object.
DrawObjectFeature	Draw a given feature of the designated object.
DrawObjectFeatureWithCurrentPen	Draw a given feature of the designated object.
DrawObjectWithCurrentPen	Draw the designated object.
DrawWithCurrentPen	Draw the designated coded element, all the objects from a layer, or all the coded elements from a given selection.
ECodedImage2	Constructs a coded image.
FindObject	Finds an object in the coded image using its coordinates.
GetObj	Random access to an object in a given layer.
GetObjCount	Returns the number of objects in a given layer.

[GetParentObject](#)

Returns a reference to the object that contains a given hole.

[RenderMask](#)

Creates a Flexible Mask from a specified layer of the encoded image.

C

ECodedImage2.ClearFeatureCache

Clears the internal cache for the computed features.

[VB6]

```
void ClearFeatureCache()  
 )
```

Remarks

This is useful to reduce memory consumption.

ECodedImage2.Draw

Draw the designated coded element, all the objects from a layer, or all the coded elements from a given selection.

[VB6]

```
void Draw(
    EDrawAdapter graphicContext,
    ECodedElement element,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void Draw(
    EDrawAdapter graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void Draw(
    EDrawAdapter graphicContext,
    Long layerIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void Draw(
    EDrawAdapter graphicContext,
    EObjectSelection selection,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void Draw(
    EDrawAdapter graphicContext,
    EObjectSelection selection,
    Long elementIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

```
void Draw(
    Long graphicContext,
    ECodedElement element,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void Draw(
    Long graphicContext,
    ERGBColor color,
    ECodedElement element,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void Draw(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void Draw(
    Long graphicContext,
    ERGBColor color,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void Draw(
    Long graphicContext,
    Long layerIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

```
void Draw(
    Long graphicContext,
    ERGBColor color,
    Long layerIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void Draw(
    Long graphicContext,
    EObjectSelection selection,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void Draw(
    Long graphicContext,
    ERGBColor color,
    EObjectSelection selection,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void Draw(
    Long graphicContext,
    EObjectSelection selection,
    Long elementIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

```
void Draw(
    Long graphicContext,
    ERGBColor color,
    EObjectSelection selection,
    Long elementIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

graphicContext

-

element

The coded element to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

layerIndex

Index of the layer of interest. If no layer index is specified, the default layer is taken into consideration. Note that this methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

selection

The selection of coded elements to draw.

elementIndex

The index in the selection of the coded element to draw. If no element index is specified, all the elements of the selection are take into consideration.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used.

ECodedImage2.DrawFeature

Draw a given feature of the designated coded element, of all the objects from a layer, or of all the coded elements from a given selection.

[VB6]

```
void DrawFeature(
    EDrawAdapter graphicContext,
    EDrawableFeature feature,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void DrawFeature(
    EDrawAdapter graphicContext,
    EDrawableFeature feature,
    Long layerIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void DrawFeature(
    EDrawAdapter graphicContext,
    EDrawableFeature feature,
    ECodedElement element,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)
```

```
void DrawFeature(
    EDRAPIAdapter graphicContext,
    EDRAPIFeature feature,
    EDRAPIObjectSelection selection,
    Long elementIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void DrawFeature(
    EDRAPIAdapter graphicContext,
    EDRAPIFeature feature,
    EDRAPIObjectSelection selection,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void DrawFeature(
    Long graphicContext,
    EDRAPIFeature feature,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void DrawFeature(
    Long graphicContext,
    ERGBColor color,
    EDRAPIFeature feature,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)
```

```
void DrawFeature(
    Long graphicContext,
    EDrawableFeature feature,
    Long layerIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void DrawFeature(
    Long graphicContext,
    ERGBColor color,
    EDrawableFeature feature,
    Long layerIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void DrawFeature(
    Long graphicContext,
    EDrawableFeature feature,
    ECodedElement element,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void DrawFeature(
    Long graphicContext,
    ERGBColor color,
    EDrawableFeature feature,
    ECodedElement element,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)
```

```
void DrawFeature(
    Long graphicContext,
    EDrawableFeature feature,
    EObjectSelection selection,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void DrawFeature(
    Long graphicContext,
    ERGBColor color,
    EDrawableFeature feature,
    EObjectSelection selection,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void DrawFeature(
    Long graphicContext,
    EDrawableFeature feature,
    EObjectSelection selection,
    Long elementIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void DrawFeature(
    Long graphicContext,
    ERGBColor color,
    EDrawableFeature feature,
    EObjectSelection selection,
    Long elementIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)
```

Parameters*graphicContext*

Graphic context on which to draw.

feature

The feature of interest.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

drawDiagonals

Specifies whether or not the diagonals of the ellipses and of the rectangles are to be drawn.

layerIndex

Index of the layer of interest. If no layer index is specified, the default layer is taken into consideration. Note that this methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

element

The coded element to draw.

selection

The selection of coded elements to draw.

elementIndex

The index in the selection of the coded element to draw. If no element index is specified, all the elements of the selection are take into consideration.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used.

The features [EDrawableFeature_FeretBox](#) and [EDrawableFeature_WeightedGravityCenter](#) are only at one's disposal when drawing selections.

ECodedImage2.DrawFeatureWithCurrentPen

Draw a given feature of the designated coded element, of all the objects from a layer, or of all the coded elements from a given selection.

[VB6]

```
void DrawFeatureWithCurrentPen(
    Long graphicContext,
    EDrawableFeature feature,
    Long layerIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void DrawFeatureWithCurrentPen(
    Long graphicContext,
    EDrawableFeature feature,
    ECodedElement element,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void DrawFeatureWithCurrentPen(
    Long graphicContext,
    EDrawableFeature feature,
    EObjectSelection selection,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)
```

```
void DrawFeatureWithCurrentPen(
    Long graphicContext,
    EDrawableFeature feature,
    EObjectSelection selection,
    Long elementIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void DrawFeatureWithCurrentPen(
    Long graphicContext,
    EDrawableFeature feature,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)
```

Parameters

graphicContext

Graphic context on which to draw.

feature

The feature of interest.

layerIndex

Index of the layer of interest. If no layer index is specified, the default layer is taken into consideration. Note that this methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

drawDiagonals

Specifies whether or not the diagonals of the ellipses and of the rectangles are to be drawn.

element

The coded element to draw.

selection

The selection of coded elements to draw.

elementIndex

The index in the selection of the coded element to draw. If no element index is specified, all the elements of the selection are take into consideration.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used.

The features [EDrawableFeature_FeretBox](#) and [EDrawableFeature_WeightedGravityCenter](#) are only at one's disposal when drawing selections.

ECodedImage2.DrawHole

Draw the designated hole.

[VB6]

```
void DrawHole(
    EDrawAdapter graphicContext,
    Long objectIndex,
    Long holeIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawHole(
    EDrawAdapter graphicContext,
    Long layerIndex,
    Long objectIndex,
    Long holeIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

```
void DrawHole(
    Long graphicContext,
    Long objectIndex,
    Long holeIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawHole(
    Long graphicContext,
    ERGBColor color,
    Long objectIndex,
    Long holeIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawHole(
    Long graphicContext,
    Long layerIndex,
    Long objectIndex,
    Long holeIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawHole(
    Long graphicContext,
    ERGBColor color,
    Long layerIndex,
    Long objectIndex,
    Long holeIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

graphicContext

Graphic context on which to draw.

objectIndex

Index of the parent object of the hole to draw.

holeIndex

Index of the hole to draw.

zoomX

Horizontal zooming factor. By default, no scaling is used.

zoomY

Vertical zooming factor. If set to **0**, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used.

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

ECodedImage2.DrawHoleFeature

Draw a given feature of the designated hole.

[VB6]

```
void DrawHoleFeature(
    EDrawAdapter graphicContext,
    EDrawableFeature feature,
    Long objectIndex,
    Long holeIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void DrawHoleFeature(
    EDrawAdapter graphicContext,
    EDrawableFeature feature,
    Long layerIndex,
    Long objectIndex,
    Long holeIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void DrawHoleFeature(
    Long graphicContext,
    EDrawableFeature feature,
    Long objectIndex,
    Long holeIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)
```

```
void DrawHoleFeature(
    Long graphicContext,
    ERGBColor color,
    EDrawableFeature feature,
    Long objectIndex,
    Long holeIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void DrawHoleFeature(
    Long graphicContext,
    EDrawableFeature feature,
    Long layerIndex,
    Long objectIndex,
    Long holeIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void DrawHoleFeature(
    Long graphicContext,
    ERGBColor color,
    EDrawableFeature feature,
    Long layerIndex,
    Long objectIndex,
    Long holeIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)
```

Parameters

graphicContext

Graphic context on which to draw.

feature

The feature of interest.

objectIndex

Index of the parent object of the hole to draw.

holeIndex

Index of the hole to draw.

zoomX

Horizontal zooming factor. By default, no scaling is used.

zoomY

Vertical zooming factor. If set to **0**, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

drawDiagonals

Specifies whether or not the diagonals of the ellipses and of the rectangles are to be drawn.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used.

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

Trying to draw the features [EDrawableFeature_FeretBox](#) and [EDrawableFeature_WeightedGravityCenter](#) will result in an exception, as they only make sense for [EObjectSelection](#).

ECodeImage2.DrawHoleFeatureWithCurrentPen

Draw a given feature of the designated hole.

[VB6]

```
void DrawHoleFeatureWithCurrentPen(
    Long graphicContext,
    EDrawableFeature feature,
    Long objectIndex,
    Long holeIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void DrawHoleFeatureWithCurrentPen(
    Long graphicContext,
    EDrawableFeature feature,
    Long layerIndex,
    Long objectIndex,
    Long holeIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)
```

Parameters

graphicContext

Graphic context on which to draw.

feature

The feature of interest.

objectIndex

Index of the parent object of the hole to draw.

holeIndex

Index of the hole to draw.

zoomX

Horizontal zooming factor. By default, no scaling is used.

zoomY

Vertical zooming factor. If set to **0**, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

drawDiagonals

Specifies whether or not the diagonals of the ellipses and of the rectangles are to be drawn.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used.

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

Trying to draw the features [EDrawableFeature_FeretBox](#) and [EDrawableFeature_WeightedGravityCenter](#) will result in an exception, as they only make sense for [EObjectSelection](#).

ECodedImage2.DrawHoleWithCurrentPen

-

[VB6]

```
void DrawHoleWithCurrentPen(
    Long graphicContext,
    Long layerIndex,
    Long objectIndex,
    Long holeIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

```
void DrawHoleWithCurrentPen(  
    Long graphicContext,  
    Long objectIndex,  
    Long holeIndex,  
    Single zoomX,  
    Single zoomY,  
    Single panX,  
    Single panY  
)
```

Parameters

graphicContext

-

layerIndex

-

objectIndex

-

holeIndex

-

zoomX

-

zoomY

-

panX

-

panY

-

ECodedImage2.DrawObject

Draw the designated object.

[VB6]

```
void DrawObject(
    EDRAutoAdapter graphicContext,
    Long objectIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawObject(
    EDRAutoAdapter graphicContext,
    Long layerIndex,
    Long objectIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawObject(
    Long graphicContext,
    Long objectIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawObject(
    Long graphicContext,
    ERGBColor color,
    Long objectIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawObject(
    Long graphicContext,
    Long layerIndex,
    Long objectIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

```
void DrawObject(
    Long graphicContext,
    ERGBColor color,
    Long layerIndex,
    Long objectIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

graphicContext

Graphic context on which to draw.

objectIndex

Index of the object to draw.

zoomX

Horizontal zooming factor. By default, no scaling is used.

zoomY

Vertical zooming factor. If set to **0**, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used.

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

ECodedImage2.DrawObjectFeature

Draw a given feature of the designated object.

[VB6]

```
void DrawObjectFeature(
    EDraWAdapter graphicContext,
    EDraWableFeature feature,
    Long objectIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void DrawObjectFeature(
    EDraWAdapter graphicContext,
    EDraWableFeature feature,
    Long layerIndex,
    Long objectIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void DrawObjectFeature(
    Long graphicContext,
    EDraWableFeature feature,
    Long objectIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)
```

```
void DrawObjectFeature(
    Long graphicContext,
    ERGBColor color,
    EDrawableFeature feature,
    Long objectIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void DrawObjectFeature(
    Long graphicContext,
    EDrawableFeature feature,
    Long layerIndex,
    Long objectIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void DrawObjectFeature(
    Long graphicContext,
    ERGBColor color,
    EDrawableFeature feature,
    Long layerIndex,
    Long objectIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)
```

Parameters

graphicContext

Graphic context on which to draw.

feature

The feature of interest.

objectIndex

Index of the object to draw.

zoomX

Horizontal zooming factor. By default, no scaling is used.

zoomY

Vertical zooming factor. If set to **0**, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

drawDiagonals

Specifies whether or not the diagonals of the ellipses and of the rectangles are to be drawn.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used.

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

Trying to draw the features [EDrawableFeature_FeretBox](#) and [EDrawableFeature_WeightedGravityCenter](#) will result in an exception, as they only make sense for [EObjectSelection](#).

ECodedImage2.DrawObjectFeatureWithCurrentPen

Draw a given feature of the designated object.

[VB6]

```

void DrawObjectFeatureWithCurrentPen(
    Long graphicContext,
    EDrawableFeature feature,
    Long objectIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void DrawObjectFeatureWithCurrentPen(
    Long graphicContext,
    EDrawableFeature feature,
    Long layerIndex,
    Long objectIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

```

Parameters

graphicContext

Graphic context on which to draw.

feature

The feature of interest.

objectIndex

Index of the object to draw.

zoomX

Horizontal zooming factor. By default, no scaling is used.

zoomY

Vertical zooming factor. If set to **0**, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

drawDiagonals

Specifies whether or not the diagonals of the ellipses and of the rectangles are to be drawn.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used.

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

Trying to draw the features [EDrawableFeature_FeretBox](#) and [EDrawableFeature_WeightedGravityCenter](#) will result in an exception, as they only make sense for [EObjectSelection](#).

ECodedImage2.DrawObjectWithCurrentPen

Draw the designated object.

[VB6]

```
void DrawObjectWithCurrentPen(
    Long graphicContext,
    Long objectIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawObjectWithCurrentPen(
    Long graphicContext,
    Long layerIndex,
    Long objectIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

graphicContext

Graphic context on which to draw.

objectIndex

Index of the object to draw.

zoomX

Horizontal zooming factor. By default, no scaling is used.

zoomY

Vertical zooming factor. If set to **0**, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used.

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

ECodedImage2.DrawWithCurrentPen

Draw the designated coded element, all the objects from a layer, or all the coded elements from a given selection.

[VB6]

```
void DrawWithCurrentPen(
    Long graphicContext,
    ECodedElement element,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

```
void DrawWithCurrentPen(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawWithCurrentPen(
    Long graphicContext,
    Long layerIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawWithCurrentPen(
    Long graphicContext,
    EObjectSelection selection,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawWithCurrentPen(
    Long graphicContext,
    EObjectSelection selection,
    Long elementIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

graphicContext

-

element

The coded element to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

layerIndex

Index of the layer of interest. If no layer index is specified, the default layer is taken into consideration. Note that this methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

selection

The selection of coded elements to draw.

elementIndex

The index in the selection of the coded element to draw. If no element index is specified, all the elements of the selection are take into consideration.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used.

ECodedImage2.ECodedImage2

Constructs a coded image.

[VB6]

```
void ECodedImage2(
    ECodedImage2 other
)
void ECodedImage2(
)
```

Parameters

other

ECodedImage2.FindObject

Finds an object in the coded image using its coordinates.

```
[VB6]  
EObject FindObject(  
    Long x,  
    Long y  
)  
  
EObject FindObject(  
    Long layerIndex,  
    Long x,  
    Long y  
)
```

Parameters

x

The X-coordinate of the object.

y

The Y-coordinate of the object.

layerIndex

The index of the layer of interest.

Remarks

If no layer index is specified, all the layers of the coded image are scanned until an object is found at these coordinates.

ECodedImage2.GetObj

Random access to an object in a given layer.

```
[VB6]
```

```

EObject GetObj(
    Long layerIndex,
    Long objectIndex
)

EObject GetObj(
    Long layerIndex,
    Long objectIndex
)

EObject GetObj(
    Long objectIndex
)

EObject GetObj(
    Long objectIndex
)

```

Parameters

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

objectIndex

The index of the object in the layer.

Remarks

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

ECodedImage2.GetObjCount

Returns the number of objects in a given layer.

[VB6]

```

Long GetObjCount(
    Long layerIndex
)

Long GetObjCount(
)

```

Parameters

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

Remarks

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

ECodedImage2.GetParentObject

Returns a reference to the object that contains a given hole.

[VB6]

```
EObject GetParentObject(  
    EHole hole  
)
```

Parameters

hole

The hole of interest.

ECodedImage2.Height

Returns the height of the coded image.

[VB6]

Height As Long

read-only

Remarks

If the continuous mode is not activated, this height corresponds to the height of the source image. If the continuous mode is activated, this value equals to the highest row index, for all the runs of all the objects in the coded image, augmented by the number of rows index that are below zero.

ECodedImage2.LayerCount

Returns the number of layers that are encoded.

[VB6]

LayerCount As Long
read-only

ECodedImage2.RenderMask

Creates a Flexible Mask from a specified layer of the encoded image.

[VB6]

```
void RenderMask(  
    EROIBW8 result  
)  
  
void RenderMask(  
    EROIBW8 result,  
    Long layerIndex,  
    Long offsetX,  
    Long offsetY  
)  
  
void RenderMask(  
    EROIBW8 result,  
    Long layerIndex  
)
```

```
void RenderMask(
    EROIBW8 result,
    Long offsetX,
    Long offsetY
)
```

Parameters*result*

The image in which the generated mask will be stored.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will serve as a source for the mask generation.

offsetX

The X-offset that must be applied to bring the zero X-coordinate in the coded image on the first column of the result image (defaults to zero).

offsetY

The Y-offset that must be applied to bring the zero Y-coordinate in the coded image on the first row of the result image (defaults to zero).

Remarks

The size of the result image will not be changed: It must be properly sized beforehand.

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

EEncodedImage2.StartY

Returns the lowest row index, for all the runs of all the objects in the coded image.

[VB6]**StartY As Long**

read-only

Remarks

The returned value will always be zero if the continuous mode is not activated.

ECodedImage2.Width

Returns the width of the coded image.

[VB6]

Width As Long

read-only

Remarks

This width corresponds in any case to the width of the source image.

EColorLookup Class

Describes a color lookup table, that is used to speed-up complex conversions between color systems.

PROPERTIES

ColorSystemIn

Input color system.

ColorSystemOut

Output color system.

IndexBits

Number of bits used for indexing the lookup table.

Interpolation

Interpolation mode.

METHODS

AdjustGainOffset	Sets a color transformation such that a gain and offset is applied separately to each color component of a target color system.
Calibrate	Sets a color transformation to recalibrate.
ConvertFromRgb	Sets a color transformation from the EColorSystem_Rgb representation to another system, as defined by EColorSystem .
ConvertToRgb	Sets a color transformation from any color system, as defined by EColorSystem , to the EColorSystem_Rgb representation.
EColorLookup	Constructs a color lookup table.
Transform	Transforms a quantized color image/pixel to another quantized color image/pixel, using the previously initialized color lookup table.
WhiteBalance	Initializes a color lookup table that can be used for color adjustment, including a global gain, gamma pre-compensation [cancellation] and white balancing.

EColorLookup.AdjustGainOffset

Sets a color transformation such that a gain and offset is applied separately to each color component of a target color system.

[VB6]

```
void AdjustGainOffset(
    EColorSystem colorSystem,
    Single gain0,
    Single offset0,
    Single gain1,
    Single offset1,
    Single gain2,
    Single offset2
)
```

Parameters

colorSystem

Target color system, as defined by [EColorSystem](#).

gain0

Gain to be applied to color component 0.

offset0

Offset to be applied to color component 0.

gain1

Gain to be applied to color component 1.

offset1

Offset to be applied to color component 1.

gain2

Gain to be applied to color component 2.

offset2

Offset to be applied to color component 2.

Remarks

The input and output color systems are both [EColorSystem_Rgb](#). To apply some transform to a color image, you initialize a color lookup once for all and use it at will in a transformation operation such as [EColorLookup::Transform](#).

Note. The offsets are specified as unquantized values.

EColorLookup.Calibrate

Sets a color transformation to recalibrate.

[VB6]

```
void Calibrate(
    EC24 Color0,
    Single x0,
    Single y0,
    Single z0,
    EC24 Color1,
    Single x1,
    Single y1,
    Single z1,
    EC24 Color2,
    Single x2,
    Single y2,
    Single z2
)

void Calibrate(
    EC24 Color0,
    Single x0,
    Single y0,
    Single z0,
    EC24 Color1,
    Single x1,
    Single y1,
    Single z1,
    EC24 Color2,
    Single x2,
    Single y2,
    Single z2,
    EC24 Color3,
    Single x3,
    Single y3,
    Single z3
)
```

```
void Calibrate(
    EC24 color,
    Single x,
    Single y,
    Single z
)
```

Parameters

Color0

Measured quantized values of a pixel of color 0.

x0

CIE XYZ tri-stimulus unquantized values corresponding to color 0.

y0

CIE XYZ tri-stimulus unquantized values corresponding to color 0.

z0

CIE XYZ tri-stimulus unquantized values corresponding to color 0.

Color1

Measured quantized values of a pixel of color 1.

x1

CIE XYZ tri-stimulus unquantized values corresponding to color 1.

y1

CIE XYZ tri-stimulus unquantized values corresponding to color 1.

z1

CIE XYZ tri-stimulus unquantized values corresponding to color 1.

Color2

Measured quantized values of a pixel of color 2.

x2

CIE XYZ tri-stimulus unquantized values corresponding to color 2.

y2

CIE XYZ tri-stimulus unquantized values corresponding to color 2.

z2

CIE XYZ tri-stimulus unquantized values corresponding to color 2.

Color3

Measured quantized values of a pixel of color 3.

x3

CIE XYZ tri-stimulus unquantized values corresponding to color 3.

y3

CIE XYZ tri-stimulus unquantized values corresponding to color 3.

z3

CIE XYZ tri-stimulus unquantized values corresponding to color 3.

color

-

x

-

y

-

z

-

Remarks

The first prototype uses 3 reference colors. The second uses 4 reference colors. To apply some transform to a color image, you initialize a color lookup once for all and use it at will in a transformation operation such as [EColorLookup::Transform](#).

EColorLookup.ColorSystemIn

Input color system.

[VB6]

ColorSystemIn As EColorSystem

read-only

Remarks

The **EColorLookup** objects keep track of the color system transformation, for consistency. When applying a transformation, the source image color system (usually [EColorSystem_Rgb](#)) must match the *input color system*; the destination image will be automatically be typed with the *output color system*. In case of a mismatch, an error message is issued. These two values are set by the lookup table initialization functions. An uninitialized lookup table has both color systems set to [EColorSystem_NoColor](#).

EColorLookup.ColorSystemOut

Output color system.

[VB6]

ColorSystemOut As EColorSystem

read-only

Remarks

The **EColorLookup** objects keep track of the color system transformation, for consistency. When applying a transformation, the source image color system (usually **EColorSystem_Rgb**) must match the *input color system*; the destination image will be automatically be typed with the *output color system*. In case of a mismatch, an error message is issued. These two values are set by the lookup table initialization functions. An uninitialized lookup table has both color systems set to **EColorSystem_NoColor**.

EColorLookup.ConvertFromRgb

Sets a color transformation from the **EColorSystem_Rgb** representation to another system, as defined by **EColorSystem**.

[VB6]

```
void ConvertFromRgb(  
    EColorSystem colorSystem  
)
```

Parameters

colorSystem

Color system, as defined by **EColorSystem**.

Remarks

The input and output color systems are respectively [EColorSystem_Rgb](#) and **colorSystem**. To apply some transform to a color image, you initialize a color lookup once for all and use it at will in a transformation operation such as [EColorLookup::Transform](#).

EColorLookup.ConvertToRgb

Sets a color transformation from any color system, as defined by [EColorSystem](#), to the [EColorSystem_Rgb](#) representation.

[VB6]

```
void ConvertToRgb(
    EColorSystem colorSystem
)
```

Parameters

colorSystem

Color system, as defined by [EColorSystem](#).

Remarks

The input and output color systems are respectively **colorSystem** and [EColorSystem_Rgb](#). To apply some transform to a color image, you initialize a color lookup once for all and use it at will in a transformation operation such as [EColorLookup::Transform](#).

EColorLookup.EColorLookup

Constructs a color lookup table.

[VB6]

```
void EColorLookup(
    EColorLookup other
)
```

```
void EColorLookup(  
    )
```

Parameters

other

-

EColorLookup.IndexBits

Number of bits used for indexing the lookup table.

[VB6]

IndexBits As Long

read-write

Remarks

Before filling in a lookup table, it is necessary to decide how many table entries it requires. The **IndexBits** property indicates how many (high-order) bits of the input components are used. The relation between **IndexBits**, the number of table entries and the corresponding table size are given below: The larger the number of entries, the more accuracy is obtained. After **IndexBits** has been changed, the lookup table needs to be recomputed.

Note. Be aware that each time a color lookup table is filled, all the entries are recomputed. When **IndexBits** equals **6**, this may take a very long time. Such large lookup tables should be computed once only. Different combinations of **IndexBits** and **Interpolation** provide a trade-off between accuracy and speed for the table pre-computation and table use.

EColorLookup.Interpolation

Interpolation mode.

[VB6]

Interpolation As Boolean

read-write

Remarks

When applying a lookup table to transform pixel values, tri-linear interpolation can be used: * when interpolation is not used, the table is looked up at the entry closest to the pixel value. This gives an accuracy equal to the value of the **IndexBits** property. On the other hand, table lookup is very fast; * when interpolation is used, the table is looked up at eight neighboring entries and an adequate average is computed. This gives full accuracy (8 bits) if the transformation is smooth enough. On the other hand, table lookup is slower.

Note. The interpolation mode may be modified at any time without the need to reinitialize the lookup table.

EColorLookup.Transform

Transforms a quantized color image/pixel to another quantized color image/pixel, using the previously initialized color lookup table.

[VB6]

```
void Transform(
    EC24 sourceImageColor,
    EC24 destinationImageColor
)

void Transform(
    EROIC24 sourceImage,
    EROIC24 destinationImage
)
```

Parameters

sourceImageColor
Input color image.
destinationImageColor
Output color image.
sourceImage

Input color image.
destinationImage
 Output color image.

EColorLookup.WhiteBalance

Initializes a color lookup table that can be used for color adjustment, including a global gain, gamma pre-compensation [cancellation] and white balancing.

[VB6]

```
void WhiteBalance(
  Single gain,
  Single gamma,
  Single balanceRed,
  Single balanceGreen,
  Single balanceBlue
)
```

Parameters

gain

Global gain to be applied to all three color components. By default, the image intensity remains unchanged.

gamma

Gamma exponent. Setting this parameter will cancel the gamma pre-compensation feature of the camera, or apply it. By default, the no gamma pre-compensation is assumed (linear response). The gamma exponent can be chosen among the predefined values

[EasyColor::CompensateNtscGamma](#)

[/EasyColor::CompensatePalGamma/EasyColor::CompensateSmpteGamma](#) (pre-compensation) or [EasyColor::NtscGamma/EasyColor::PalGamma/EasyColor::SmpteGamma](#) (pre-compensation cancellation), or be user-defined.

balanceRed

Color values to be used for white balance. These parameters should be set to the measured average values of a white (or gray) pixels area, allowing the white balance to adjust pre-component gains appropriately. Use function [EasylImage::PixelAverage](#) to obtain them. By default, no white balancing is performed.

balanceGreen

Color values to be used for white balance. These parameters should be set to the measured average values of a white (or gray) pixels area, allowing the white balance to adjust pre-component gains appropriately. Use function [EasylImage::PixelAverage](#) to obtain them. By default, no white balancing is performed.

balanceBlue

Color values to be used for white balance. These parameters should be set to the measured average values of a white (or gray) pixels area, allowing the white balance to adjust pre-component gains appropriately. Use function [EasylImage::PixelAverage](#) to obtain them. By default, no white balancing is performed.

Remarks

To apply some transform to a color image, you initialize the color lookup once for all and use it at will with [EColorLookup::Transform](#) or [EasyColor::TransformBayer](#) operation.

EColorRangeThresholdSegmenter Class

Segments an image using a double threshold on a color image.

Remarks

This segmenter is applicable to [EROIC24](#) RGB color images. It produces coded images with two layers: The White layer (usually, with index 1) contains the unmasked pixels that belong to the cube of the RGB space that spans the low threshold point to the high threshold point; and the Black layer (usually, with index 0) contains the remaining unmasked pixels.

Base Class: [ETwoLayersImageSegmenter](#)

PROPERTIES

HighThreshold

Value of the high threshold.

LowThreshold

Value of the low threshold.

EColorRangeThresholdSegmenter.HighThreshold

Value of the high threshold.

[VB6]

HighThreshold As EC24

read-write

EColorRangeThresholdSegmenter.LowThreshold

Value of the low threshold.

[VB6]

LowThreshold As EC24

read-write

EColorSingleThresholdSegmenter Class

Segments an image using a single threshold on a color image.

Remarks

This segmenter is applicable to [EROIC24](#) RGB color images. It produces coded images with two layers: The White layer (usually, with index 1) contains the unmasked pixels that belong to the cube of the RGB space defined by the threshold point and the white point **(255,255,255)**; and the Black layer (usually, with index 0) contains the remaining unmasked pixels.

Base Class: [ETwoLayersImageSegmenter](#)

PROPERTIES**Threshold**

| E Value of the threshold.

C

ColorSingleThresholdSegmenter.Threshold

Value of the threshold.

[VB6]

Threshold As EC24

read-write

EColorVector Class**Base Class:** EVector**PROPERTIES****RawDataPtr**

| M Pointer to the vector data.

E

THODS**AddElement**

| Append (adds at the tail) an element to the vector.

EColorVector	-
GetElement	Returns the vector element at the given index.
operator[]	Gives access to the vector element at the given index.
operator=	Copies all the data from another EColorVector object into the current EColorVector object
SetElement	E Modifies the vector element at the given index by the given value. C

olorVector.AddElement

Appends (adds at the tail) an element to the vector.

[VB6]

```
void AddElement(
    EColor element
)
```

Parameters

element

The element to be added.

EColorVector.EColorVector

```
[VB6]
void EColorVector(
)
void EColorVector(
    Long un32MaxElements
)
void EColorVector(
    EColorVector other
)
```

Parameters

un32MaxElements

other

EColorVector.GetElement

Returns the vector element at the given index.

```
[VB6]
EColor GetElement(
    Long index
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded) of the element to be accessed.

Remarks

If the given index is outside the bounds of the vector, the error code [EError_Parameter1OutOfRange](#) is set.

EColorVector.operator[]

Gives access to the vector element at the given index.

[VB6]

```
EColor operator[](  
    Long index  
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded) of the element to be accessed.

EColorVector.operator=

Copies all the data from another EColorVector object into the current EColorVector object

[VB6]

```
EColorVector operator=(  
    EColorVector other  
)
```

Parameters

other

EColorVector object to be copied

EColorVector.RawDataPtr

Pointer to the vector data.

[VB6]

RawDataPtr As Long

read-only

EColorVector.SetElement

Modifies the vector element at the given index by the given value.

[VB6]

```
void SetElement(  
    Long index,  
    EColor value  
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded), of the element to be modified.

value

The new value for the element.

Remarks

If the given index is outside the bounds of the vector, the error code [EError_Parameter1OutOfRange](#) is set.

EDepthMap Class

Represents a depth map.

Base Class: [EPixelContainer](#)

Derived Class(es): [EDepthMap8](#) [EDepthMap16](#) [EDepthMap32f](#)

EDepthMap16 Class

Represents a depth map with a 16-bit pixel representation.

Base Class: [EDepthMap](#)

PROPERTIES

Precision

Fixed-point precision of the pixel values. The precision is the number of bits after the decimal point. Starts from 0 (no fractional part) and ends to 16.

RootROI

Root ROI.

Type

Pixel accessor type.

UndefinedValue

M Return the Undefined value. That value is used to set pixel with no valid depth value.

E

THODS

AsEImageBW16

[EDepthMap16](#)

Casts the depth map into an EImage to use with 2D eVision tools.

[GetPixel](#)

Constructs a [EDepthMap16](#).

Gets a pixel.

[Serialize](#)

Serializes the object.

[SetPixel](#)

Sets a pixel.

D

epthMap16.AsEImageBW16

Casts the depth map into an EImage to use with 2D eVision tools.

[VB6]

```
EImageBW16 AsEImageBW16(  
)
```

EDepthMap16.EDepthMap16

Constructs a [EDepthMap16](#).

```
[VB6]
void EDepthMap16(
    )
void EDepthMap16(
    Long width,
    Long height
)
void EDepthMap16(
    EDepthMap16 other
)
```

Parameters

width

Width of the depth map.

height

Height of the depth map.

other

-

EDepthMap16.GetPixel

Gets a pixel.

```
[VB6]
EDepth16 GetPixel(
    Long x,
    Long y
)
```

Parameters

x

Column of the pixel.

y

Row of the pixel.

EDepthMap16.Precision

Fixed-point precision of the pixel values. The precision is the number of bits after the decimal point. Starts from 0 (no fractional part) and ends to 16.

[VB6]

Precision As Long

read-write

EDepthMap16.RootROI

Root ROI.

[VB6]

RootROI As EDepthMapROI16

read-only

EDepthMap16.Serialize

Serializes the object.

[VB6]

```
void Serialize(  
    ESerializer serializer  
)
```

Parameters

serializer

EDepthMap16.SetPixel

Sets a pixel.

```
[VB6]  
void SetPixel(  
    EDepth16 value,  
    Long x,  
    Long y  
)
```

Parameters

value

Value of the pixel.

x

Column of the pixel.

y

Row of the pixel.

EDepthMap16.Type

Pixel accessor type.

```
[VB6]  
Type As EImageType  
read-only
```

EDepthMap16.UndefinedValue

Return the Undefined value. That value is used to set pixel with no valid depth value.

[VB6]

UndefinedValue As EDepth16

read-only

EDepthMap32f Class

Represents a depth map with an 32-bit floating point pixel representation.

Base Class: [EDepthMap](#)

PROPERTIES

[RootROI](#)

Root ROI.

[Type](#)

Pixel accessor type.

[UndefinedValue](#)

M Return the Undefined value. That value is used to set pixel with no valid depth value.

E

THODS

[EDepthMap32f](#)

Constructs a [EDepthMap32f](#).

[GetPixel](#) Gets a pixel.

[Serialize](#) Serializes the object.

[SetPixel](#) Sets a pixel.

D

epthMap32f.EDepthMap32f

Constructs a [EDepthMap32f](#).

[VB6]

```
void EDepthMap32f()
)
void EDepthMap32f(
    Long width,
    Long height
)
void EDepthMap32f(
    EDepthMap32f other
)
```

Parameters

width

Width of the depyh map.

height

Height of the depth map.

other

EDepthMap32f.GetPixel

Gets a pixel.

[VB6]

```
EDepth32f GetPixel(  
    Long x,  
    Long y  
)
```

Parameters

x

Column of the pixel.

y

Row of the pixel.

EDepthMap32f.RootROI

Root ROI.

[VB6]

```
RootROI As EDepthMapROI32f  
read-only
```

EDepthMap32f.Serialize

Serializes the object.

[VB6]

```
void Serialize(  
    ESerializer serializer  
)
```

Parameters

serializer

-

EDepthMap32f.SetPixel

Sets a pixel.

[VB6]

```
void SetPixel(  
    EDepth32f value,  
    Long x,  
    Long y  
)
```

Parameters

value

Value of the pixel.

x

Column of the pixel.

y

Row of the pixel.

EDepthMap32f.Type

Pixel accessor type.

[VB6]

Type As EIImageType

read-only

EDepthMap32f.UndefinedValue

Return the Undefined value. That value is used to set pixel with no valid depth value.

[VB6]

UndefinedValue As EDepth32f

read-only

EDepthMap8 Class

Represents a depth map with an 8-bit pixel representation.

Base Class: [EDepthMap](#)

PROPERTIES

RootROI

Root ROI.

Type

Pixel accessor type.

UndefinedValue

Return the Undefined value. That value is used to set pixel with no valid depth value.

METHODS

[AsEImageBW8](#)

Casts the depth map into an EImage to use with 2D eVision tools.

[EDepthMap8](#)

Constructs a [EDepthMap8](#).

[GetPixel](#)

Gets a pixel.

[Serialize](#)

Serializes the object.

[SetPixel](#)

Sets a pixel.

D

DepthMap8.AsEImageBW8

Casts the depth map into an EImage to use with 2D eVision tools.

[VB6]

```
EImageBW8 AsEImageBW8 (
)
```

EDepthMap8.EDepthMap8

Constructs a [EDepthMap8](#).

```
[VB6]
void EDepthMap8 (
)
void EDepthMap8 (
    Long width,
    Long height
)
void EDepthMap8 (
    EDepthMap8 other
)
```

Parameters

width

Width of the depyh map.

height

Height of the depth map.

other

-

EDepthMap8.GetPixel

Gets a pixel.

```
[VB6]
EDepth8 GetPixel(
    Long x,
    Long y
)
```

Parameters

x

Column of the pixel.

y

Row of the pixel.

EDepthMap8.RootROI

Root ROI.

[VB6]

RootROI As EDepthMapROI8

read-only

EDepthMap8.Serialize

Serializes the object.

[VB6]

```
void Serialize(  
    ESerializer serializer  
)
```

Parameters

serializer

-

EDepthMap8.SetPixel

Sets a pixel.

[VB6]

```
void SetPixel(  
    EDepth8 value,  
    Long x,  
    Long y  
)
```

Parameters

value

Value of the pixel.

x

Column of the pixel.

y

Row of the pixel.

EDepthMap8.Type

Pixel accessor type.

[VB6]

Type As EImageType

read-only

EDepthMap8.UndefinedValue

Return the Undefined value. That value is used to set pixel with no valid depth value.

[VB6]

UndefinedValue As EDepth8

read-only

EDepthMapROI Class

Represents a ROI on a depth map.

Base Class: [EROI](#)

Derived Class(es): [EDepthMapROI8](#) [EDepthMapROI16](#) [EDepthMapROI32f](#)

EDepthMapROI16 Class

Represents a ROI on a depth map with a 16-bit pixel representation.

Base Class: [EDepthMapROI](#)

PROPERTIES

Precision Fixed-point precision of the pixel values.

RootROI Root ROI.

Type Pixel accessor type.

UndefinedValue **M** Undefined value.

E

THODS

AsEROIBW16 Casts the contents of the ROI into an ElImage to use with 2D eVision tools.

Attach	Attaches the ROI to another ROI of the same type.
CreateNew	Factory.
EDepthMapROI16	Constructs a EDepthMapROI16 .
GetPixel	Gets a pixel.
Serialize	Serializes the object.
SetPixel	Sets a pixel.

D

DepthMapROI16.AsEROIBW16

Casts the contents of the ROI into an ElImage to use with 2D eVision tools.

```
[VB6]  
EROIBW16 AsEROIBW16(  
)
```

EDepthMapROI16.Attach

Attaches the ROI to another ROI of the same type.

[VB6]

```
void Attach(  
    EDepthMapROI16 parent  
)
```

Parameters

parent

-

EDepthMapROI16.CreateNew

Factory.

[VB6]

```
EROI CreateNew(  
)
```

EDepthMapROI16.EDepthMapROI16

Constructs a [EDepthMapROI16](#).

[VB6]

```
void EDepthMapROI16(  
)  
  
void EDepthMapROI16(  
    Long width,  
    Long height  
)
```

```
void EDepthMapROI16(
    EDepthMapROI16 other
)
```

Parameters

width

Width of the depth map ROI.

height

Height of the depth map ROI.

other

-

EDepthMapROI16.GetPixel

Gets a pixel.

[VB6]

```
EDepth16 GetPixel(
    Long x,
    Long y
)
```

Parameters

x

Column of the pixel.

y

Row of the pixel.

EDepthMapROI16.Precision

Fixed-point precision of the pixel values.

[VB6]

Precision As Long

read-only

EDepthMapROI16.RootROI

Root ROI.

[VB6]

RootROI As EDepthMapROI16

read-only

EDepthMapROI16.Serialize

Serializes the object.

[VB6]

```
void Serialize(  
    ESerializer serializer  
)
```

Parameters

serializer

-

EDepthMapROI16.SetPixel

Sets a pixel.

[VB6]

```
void SetPixel(  
    EDepth16 value,  
    Long x,  
    Long y  
)
```

Parameters

value

Value of the pixel.

x

Column of the pixel.

y

Row of the pixel.

EDepthMapROI16.Type

Pixel accessor type.

[VB6]

Type As EImageType

read-only

EDepthMapROI16.UndefinedValue

Undefined value.

[VB6]

UndefinedValue As EDepth16

read-only

EDepthMapROI32f Class

Represents a ROI on a depth map with an 32-bit floating point pixel representation.

Base Class: [EDepthMapROI](#)

PROPERTIES

[RootROI](#) Root ROI.

[Type](#) Pixel accessor type.

[UndefinedValue](#) **M** Undefined value.

E

THODS

[Attach](#) Attaches the ROI to another ROI of the same type.

[CreateNew](#) Factory.

[EDepthMapROI32f](#) Constructs a [EDepthMapROI32f](#).

[GetPixel](#) Gets a pixel.

Serialize

Serializes the object.

SetPixel

Sets a pixel.

D

DepthMapROI32f.Attach

Attaches the ROI to another ROI of the same type.

[VB6]

```
void Attach(  
    EDepthMapROI32f parent  
)
```

Parameters

parent

-

EDepthMapROI32f.CreateNew

Factory.

[VB6]

```
EROI CreateNew(  
)
```

EDepthMapROI32f.EDepthMapROI32f

Constructs a [EDepthMapROI32f](#).

```
[VB6]  
void EDepthMapROI32f()  
)  
  
void EDepthMapROI32f(  
Long width,  
Long height  
)  
  
void EDepthMapROI32f(  
EDepthMapROI32f other  
)
```

Parameters

width

Width of the depth map ROI.

height

Height of the depth map ROI.

other

-

EDepthMapROI32f.GetPixel

Gets a pixel.

```
[VB6]  
EDepth32f GetPixel(  
Long x,  
Long y  
)
```

Parameters*x*

Column of the pixel.

y

Row of the pixel.

EDepthMapROI32f.RootROI

Root ROI.

[VB6]

```
RootROI As EDepthMapROI32f  
read-only
```

EDepthMapROI32f.Serialize

Serializes the object.

[VB6]

```
void Serialize(  
    ESerializer serializer  
)
```

Parameters*serializer*

-

EDepthMapROI32f.SetPixel

Sets a pixel.

[VB6]

```
void SetPixel(  
    EDepth32f value,  
    Long x,  
    Long y  
)
```

Parameters

value

Value of the pixel.

x

Column of the pixel.

y

Row of the pixel.

EDepthMapROI32f.Type

Pixel accessor type.

[VB6]

Type As EImageType

read-only

EDepthMapROI32f.UndefinedValue

Undefined value.

[VB6]

UndefinedValue As EDepth32f

read-only

EDepthMapROI8 Class

Represents a ROI on a depth map with a 16-bit pixel representation.

Base Class: [EDepthMapROI](#)

PROPERTIES

RootROI Root ROI.

Type Pixel accessor type.

UndefinedValue **M** Undefined value.

E

THODS

AsEROIBW8 Casts the contents of the ROI into an ElImage to use with 2D eVision tools.

Attach Attaches the ROI to another ROI of the same type.

CreateNew Factory.

EDepthMapROI8 Constructs a [EDepthMapROI8](#).

[GetPixel](#) Gets a pixel.

[Serialize](#) Serializes the object.

[SetPixel](#) Sets a pixel.

D

EDepthMapROI8.AsEROIBW8

Casts the contents of the ROI into an ElImage to use with 2D eVision tools.

[VB6]

```
EROIBW8 AsEROIBW8()  
)
```

EDepthMapROI8.Attach

Attaches the ROI to another ROI of the same type.

[VB6]

```
void Attach(  
    EDepthMapROI8 parent  
)
```

Parameters

parent

EDepthMapROI8.CreateNew

Factory.

[VB6]

```
EROI CreateNew(  
    )
```

EDepthMapROI8.EDepthMapROI8

Constructs a [EDepthMapROI8](#).

[VB6]

```
void EDepthMapROI8(  
    )  
  
void EDepthMapROI8(  
    Long width,  
    Long height  
    )  
  
void EDepthMapROI8(  
    EDepthMapROI8 other  
    )
```

Parameters

width

Width of the depth map ROI.

height

Height of the depth map ROI.

other

EDepthMapROI8.GetPixel

Gets a pixel.

[VB6]

```
EDepth8 GetPixel(  
    Long x,  
    Long y  
)
```

Parameters

x

Column of the pixel.

y

Row of the pixel.

EDepthMapROI8.RootROI

Root ROI.

[VB6]

```
RootROI As EDepthMapROI8  
read-only
```

EDepthMapROI8.Serialize

Serializes the object.

[VB6]

```
void Serialize(  
    ESerializer serializer  
)
```

Parameters

serializer

-

EDepthMapROI8.SetPixel

Sets a pixel.

[VB6]

```
void SetPixel(  
    EDepth8 value,  
    Long x,  
    Long y  
)
```

Parameters

value

Value of the pixel.

x

Column of the pixel.

y

Row of the pixel.

EDepthMapROI8.Type

Pixel accessor type.

[VB6]

Type As EIImageType

read-only

EDepthMapROI8.UndefinedValue

Undefined value.

[VB6]

UndefinedValue As EDepth8

read-only

EException Class

Holds the exception information, that is the code and the description of the error that has thrown the exception.

Remarks

Each time an Open eVision error occurs, an exception is thrown. Exceptions feature an error code and a description. To catch a potentially arising exception, the function call is included in a try-catch block.

PROPERTIES

Error

-

METHODS

EException

-

operator=

-

What

E Returns the description of the error that has thrown the exception.

E

xception.EException

[VB6]

```
void EException()
)

void EException(
    EError error
)

void EException(
    EException other
)

void EException(
    String message
)
```

Parameters

error

-

other

-

message

EException.Error

[VB6]

Error As EError

read-write

EException.operator=

[VB6]

```
EException operator=(  
    EException other  
)
```

Parameters

other

EException.What

Returns the description of the error that has thrown the exception.

[VB6]

```
String What()  
)
```

EFilePointerSerializer Class

-

Base Class: [ESerializer](#)

PROPERTIES

Writing

M Returns **TRUE** if the [ESerializer](#) object has been created for writing and **FALSE** otherwise.

E

THODS

Close

E Closes the file associated with the [ESerializer](#) object.

F

EFilePointerSerializer.Close

Closes the file associated with the [ESerializer](#) object.

[VB6]

```
void Close()  
)
```

EFilePointerSerializer.Writing

Returns **TRUE** if the [ESerializer](#) object has been created for writing and **FALSE** otherwise.

[VB6]

Writing As Boolean

read-only

EFileSerializer Class

Base Class: [ESerializer](#)

PROPERTIES

[Writing](#)

M Returns **TRUE** if the [ESerializer](#) object has been created for writing and **FALSE** otherwise.

E

THODS

[Close](#)

E Closes the file associated with the [ESerializer](#) object.

F

ileSerializer.Close

Closes the file associated with the [ESerializer](#) object.

```
[VB6]  
void Close()  
)
```

EFileSerializer.Writing

Returns **TRUE** if the [ESerializer](#) object has been created for writing and **FALSE** otherwise.

```
[VB6]  
Writing As Boolean  
read-only
```

EFloatRange Class

Represents a range of floating point values.

PROPERTIES

[LowerBound](#) Lower bound of the range.

[UpperBound](#) **M** Upper bound of the range.

E

THODS

[EFloatRange](#) Creates an [EFloatRange](#) object.

[IsInRange](#) Checks if a value is inside the range.

[SetBounds](#) Sets the bounds of the range.

[SetFromBaseAndTolerance](#) Sets the bounds of the range from a base value and a tolerance from this base value.

F

loatRange.EFloatRange

Creates an [EFloatRange](#) object.

[VB6]

```
void EFloatRange()
)
void EFloatRange(
    Single min,
    Single max
)
```

Parameters

min

Lower bound of the range.

max

Upper bound of the range.

EFloatRange.IsInRange

Checks if a value is inside the range.

[VB6]

```
Boolean IsInRange(  
    Single value,  
    Boolean lowerBoundInclusive,  
    Boolean upperBoundInclusive  
)
```

Parameters

value

Value to test.

lowerBoundInclusive

Indicates if the lower bound should be considered inside the range (true) or outside (false).

upperBoundInclusive

Indicates if the upper bound should be considered inside the range (true) or outside (false).

EFloatRange.LowerBound

Lower bound of the range.

[VB6]

LowerBound As Single

read-only

EFloatRange.SetBounds

Sets the bounds of the range.

[VB6]

```
void SetBounds(
    Single min,
    Single max
)
```

Parameters

min

Lower bound of the range.

max

Upper bound of the range.

EFloatRange.SetFromBaseAndTolerance

Sets the bounds of the range from a base value and a tolerance from this base value.

[VB6]

```
void SetFromBaseAndTolerance(
    Single baseValue,
    Single tolerance
)
```

Parameters

baseValue

Base value.

tolerance

Tolerance around the base value. Must be positive.

Remarks

The range will be set with a lower bound of (base - tolerance) and an upper bound of (base + tolerance).

EFloatRange.UpperBound

Upper bound of the range.

[VB6]

UpperBound As Single

read-only

EFoundPattern Class

Represents a single instance of the pattern in the search field, as returned by the EasyFind finding process.

Remarks

[EPatternFinder::Find](#) returns a collection of instances of this class. An EFoundPattern object represents one found instance, with all the needed information about it.

PROPERTIES

Angle

Angle of the found instance, in the current angle unit.

Center

Reference point of the instance.

DrawBoundingBox

Flag indicating if the [EFoundPattern::Draw](#) method must draw the bounding box of the **FoundPattern**.

DrawCenter

Flag indicating if the [EFoundPattern::Draw](#) method must draw the center of the **FoundPattern**.

DrawFeaturePoints

Flag indicating if the [EFoundPattern::Draw](#) method must draw the feature points of the **EFoundPattern** object.

Quadrangle	Returns the corners of the bounding box of the found pattern.
Scale	Scaling factor of the found pattern, in units (not percents).
Score	M Matching score of the found pattern, in units (not percents). E
THODS	
Draw	Draws the found pattern, in image coordinates.
DrawWithCurrentPen	Draws the found pattern, in image coordinates.
EFoundPattern	Constructs a EFoundPattern object.
operator!=	-
operator=	Copies all the data from another EFoundPattern object into the current EFoundPattern object
operator==	-

EFoundPattern.Angle

Angle of the found instance, in the current angle unit.

[VB6]

Angle As Single

read-only

Remarks

Read-only. This returned value is always comprised in the range [- a half turn, + a half turn].

EFoundPattern.Center

Reference point of the instance.

[VB6]

Center As EPoint

read-only

Remarks

By default, this is its center. If the property [EPatternFinder::Pivot](#) has been changed in the [EPatternFinder](#), the point returns the pivot in the instance.

EFoundPattern.Draw

Draws the found pattern, in image coordinates.

[VB6]

```
void Draw(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void Draw(
    Long graphicContext,
    ERGBColor color,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void Draw(
    EDrawAdapter graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

graphicContext

Handle to the device context of the destination window.

zoomX

Horizontal zooming factor.

zoomY

Vertical zooming factor. If set to **0**, the horizontal zooming factor will be used for isotropic zooming.

panX

Horizontal panning factor.

panY

Vertical panning factor.

color

The color in which to draw the overlay.

Remarks

This method draws different features of the EFoundPattern, according to the value of properties `EFoundPattern::DrawFeaturePoints`, `EFoundPattern::DrawCenter`, `EFoundPattern::DrawBoundingBox`. The `zoomX`, `zoomY`, `panX` and `panY` parameters can be used to scale and/or translate the drawing operations.

EFoundPattern.DrawBoundingBox

Flag indicating if the `EFoundPattern::Draw` method must draw the bounding box of the **FoundPattern**.

[VB6]

```
DrawBoundingBox As Boolean  
read-write
```

Remarks

The default value is **TRUE**.

EFoundPattern.DrawCenter

Flag indicating if the `EFoundPattern::Draw` method must draw the center of the **FoundPattern**.

[VB6]

```
DrawCenter As Boolean  
read-write
```

Remarks

The default value is **TRUE**.

EFoundPattern.DrawFeaturePoints

Flag indicating if the [EFoundPattern::Draw](#) method must draw the feature points of the [EFoundPattern](#) object.

[VB6]

```
DrawFeaturePoints As Boolean  
read-write
```

Remarks

The default value is **FALSE**.

EFoundPattern.DrawWithCurrentPen

Draws the found pattern, in image coordinates.

[VB6]

```
void DrawWithCurrentPen(  
    Long graphicContext,  
    Single zoomX,  
    Single zoomY,  
    Single panX,  
    Single panY  
)
```

Parameters

graphicContext

Handle to the device context of the destination window.

zoomX

Horizontal zooming factor.

zoomY

Vertical zooming factor. If set to **0**, the horizontal zooming factor will be used for isotropic zooming.

panX

Horizontal panning factor.

panY

Vertical panning factor.

Remarks

This method draws different features of the EFoundPattern, according to the value of properties [EFoundPattern::DrawFeaturePoints](#), [EFoundPattern::DrawCenter](#), [EFoundPattern::DrawBoundingBox](#). The **zoomX**, **zoomY**, **panX** and **panY** parameters can be used to scale and/or translate the drawing operations.

EFoundPattern.EFoundPattern

Constructs a EFoundPattern object.

```
[VB6]
void EFoundPattern(
)
void EFoundPattern(
    EFoundPattern other
)
```

Parameters*other*

EFoundPattern object to be copied

EFoundPattern.operator!=

-

```
[VB6]
```

```
Boolean operator!=  
EFoundPattern fnPat  
)
```

Parameters

fnPat

EFoundPattern.operator=

Copies all the data from another EFoundPattern object into the current EFoundPattern object

[VB6]

```
EFoundPattern operator=(  
EFoundPattern other  
)
```

Parameters

other

EFoundPattern object to be copied

EFoundPattern.operator==

-

```
Boolean operator==(  
EFoundPattern fnPat  
)
```

Parameters

fndPat

EFoundPattern.Quadrangle

Returns the corners of the bounding box of the found pattern.

[VB6]

Quadrangle As EQuadrangle

read-only

EFoundPattern.Scale

Scaling factor of the found pattern, in units (not percents).

[VB6]

Scale As Single

read-only

EFoundPattern.Score

Matching score of the found pattern, in units (not percents).

[VB6]

Score As Single

read-only

Remarks

The matching score range is **[-1.0..1.0]**.

EFrame Class

Represents a geometrical frame of reference as well as the parameters needed to transform from/to local and global coordinates. It contains a point and an angle and serves as a base class for geometrical elements.

Base Class: [EPoint](#)

Derived Class(es): [ECircle](#) [ELine](#) [ERectangle](#) [EWedge](#)

PROPERTIES

Angle Orientation of the frame.

CenterX Abscissa of the origin point of the frame.

CenterY Ordinate of the origin point of the frame.

Scale **M** Horizontal sensor resolution, in pixels per unit.

E

THODS

[CopyTo](#)

	Copies all the data from the current EFrame object into another EFrame object, and returns it.
EFrame	Constructs a EFrame object.
GlobalToLocal	Transforms a geometrical element from global to local coordinates (world to local).
LocalToGlobal	Transforms a geometrical element from local to global coordinates (local to world).
operator=	Copies all the data from another EFrame object into the current EFrame object.
F	

rame.Angle

Orientation of the frame.

[VB6]

Angle As Single

read-write

EFrame.CenterX

Abscissa of the origin point of the frame.

[VB6]

CenterX As Single

read-only

EFrame.CenterY

Ordinate of the origin point of the frame.

[VB6]

CenterY As Single

read-only

EFrame.CopyTo

Copies all the data from the current EFrame object into another EFrame object, and returns it.

[VB6]

```
EFrame CopyTo(  
    EFrame other  
)
```

Parameters

other

Pointer to the EFrame object in which the current EFrame object data have to be copied.

Remarks

In case of a **NULL** pointer, a new EFrame object will be created and returned.

EFrame.EFrame

Constructs a EFrame object.

[VB6]

```
void EFrame()
)

void EFrame(
    Single centerX,
    Single centerY,
    Single angle,
    Single scale
)

void EFrame(
    EPoint center,
    Single angle,
    Single scale
)

void EFrame(
    EFrame frame
)
```

Parameters

centerX

Abscissa of the origin point of the frame.

centerY

Ordinate of the origin point of the frame.

angle

Orientation of the frame.

scale

Horizontal sensor resolution.

center

Coordinates of the origin point of the frame.

frame

Pre-existing EFrame object used by the copy constructor.

EFrame.GlobalToLocal

Transforms a geometrical element from global to local coordinates (world to local).

```
[VB6]  
EPoint GlobalToLocal(  
    EPoint global  
)  
  
EFrame GlobalToLocal(  
    EFrame global  
)
```

Parameters

global

The element, expressed in global coordinates.

EFrame.LocalToGlobal

Transforms a geometrical element from local to global coordinates (local to world).

```
[VB6]  
EPoint LocalToGlobal(  
    EPoint local  
)  
  
EFrame LocalToGlobal(  
    EFrame local  
)  
  
ELine LocalToGlobal(  
    ELine local  
)  
  
ECircle LocalToGlobal(  
    ECircle local  
)
```

Parameters

local

The element, expressed in local coordinates.

EFrame.operator=

Copies all the data from another EFrame object into the current EFrame object.

[VB6]

```
EFrame operator=(  
    EFrame frame  
)
```

Parameters

frame

EFrame object to be copied.

EFrame.Scale

Horizontal sensor resolution, in pixels per unit.

[VB6]

```
Scale As Single  
read-write
```

EFrameShape Class

Manages a complete context for measuring frame shapes.

Remarks

This class allows the grouping of several gauges or other frames.

Base Class: [EShape](#)

PROPERTIES

Angle	-
Center	Origin point coordinates of the frame.
CenterX	-
CenterY	-
Scale	-
SizeX	Frame X-axis length.
SizeY	Frame Y-axis length.
Type	M Shape type. E

THODS

Closest	Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use EShape::ClosestShape .
---------	--

[CopyTo](#)

-

[Drag](#)

Moves a handle to a new position and updates the position parameters of the shape.

[Draw](#)

Draws a graphical representation of a shape, as defined by [EDrawingMode](#).

[DrawWithCurrentPen](#)

Draws a graphical representation of a shape, as defined by [EDrawingMode](#).

[EFrameShape](#)

Creates a frame shape measurement context.

[HitTest](#)

Checks whether the cursor is positioned over a handle (**TRUE**) or not (**FALSE**).

[operator=](#)

-

[Set](#)

Sets the coordinates of the origin point and the orientation of the frame.

[SetCenterXY](#)

Sets the origin point coordinates of the frame.

[SetSize](#)

Sets the frame size.

EFrameShape.Angle

-

[VB6]

Angle As Single

read-write

EFrameShape.Center

Origin point coordinates of the frame.

[VB6]

Center As EPoint

read-write

EFrameShape.CenterX

-

[VB6]

CenterX As Single

read-only

EFrameShape.CenterY

-

[VB6]

CenterY As Single

read-only

EFrameShape.Closest

Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use [EShape::ClosestShape](#).

[VB6]

```
void Closest(  
    )
```

EFrameShape.CopyTo

-

[VB6]

```
EFrameShape CopyTo(  
    EFrameShape other,  
    Boolean recursive  
)
```

Parameters

other

-

recursive

TRUE if the daughter shapes have to be copied as well, **FALSE** otherwise.

EFrameShape.Drag

Moves a handle to a new position and updates the position parameters of the shape.

[VB6]

```
void Drag(
    Long cursorX,
    Long cursorY
)
```

Parameters

cursorX

Current cursor coordinates.

cursorY

Current cursor coordinates.

EFrameShape.Draw

Draws a graphical representation of a shape, as defined by [EDrawingMode](#).

[VB6]

```
void Draw(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

```

void Draw(
    Long graphicContext,
    ERGBColor color,
    EDrawingMode drawingMode,
    Boolean daughters
)

void Draw(
    EDrawAdapter graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)

```

Parameters*graphicContext*

Handle of the device context on which to draw.

*drawingMode*Indicates how the shape must be displayed, as defined by [EDrawingMode](#).*daughters***TRUE** if the daughter shapes are to be displayed also.*color*

The color in which to draw the overlay.

EFrameShape.DrawWithCurrentPen

Draws a graphical representation of a shape, as defined by [EDrawingMode](#).

[VB6]

```

void DrawWithCurrentPen(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)

```

Parameters*graphicContext*

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by [EDrawingMode](#).

daughters

TRUE if the daughters gauges are to be displayed also.

EFrameShape.EFrameShape

Creates a frame shape measurement context.

[VB6]

```
void EFrameShape()
)

void EFrameShape(
    EFrameShape frameShape
)
```

Parameters

frameShape

Pre-existing EFrameShape object used by the copy constructor.

Remarks

With the default constructor, all parameters are initialized to their respective default value. With the copy constructor, the constructed frame shape measurement context is based on a pre-existing EFrameShape object. By default, the daughter shapes are also copied. Use the [EFrameShape::CopyTo](#) method to disable explicitly the daughter shapes copy.

EFrameShape.HitTest

Checks whether the cursor is positioned over a handle (**TRUE**) or not (**FALSE**).

[VB6]

```
Boolean HitTest(  
    Boolean daughter  
)
```

Parameters

daughter

TRUE if the daughters shapes handles have to be considered as well.

EFrameShape.operator=

-
[VB6]

```
EFrameShape operator=(  
    EFrameShape other  
)
```

Parameters

other

Remarks

By default, the daughter shapes are also copied. Use the [EFrameShape::CopyTo](#) method to disable explicitly the daughter shapes copy.

EFrameShape.Scale

-
[VB6]

Scale As Single

read-write

EFrameShape.Set

Sets the coordinates of the origin point and the orientation of the frame.

[VB6]

```
void Set(  
    EPoint center,  
    Single angle,  
    Single scale  
)
```

Parameters

center

Coordinates of the origin point of the frame. The default value is **(0,0)**.

angle

Rotation angle of the frame. The default value is **0**.

scale

Horizontal sensor resolution, in pixels per unit

EFrameShape.SetCenterXY

Sets the origin point coordinates of the frame.

[VB6]

```
void SetCenterXY(
    Single centerX,
    Single centerY
)
```

Parameters*centerX*Abscissa of the origin point of the frame. Default value is **0**.*centerY*Ordinate of the origin point of the frame. Default value is **0**.

EFrameShape.GetSize

Sets the frame size.

[VB6]

```
void SetSize(
    Single sizeX,
    Single sizeY
)
```

Parameters*sizeX*Frame X-axis length. The default value is **100**.*sizeY*

Frame Y-axis length. By default, both axes have the same length.

Remarks

By default, both frame axis value are set to **100**, which means 100 pixels when the field of view is not calibrated and 100 "units" in case of a calibrated field of view.

EFrameShape.SizeX

Frame X-axis length.

[VB6]

SizeX As Single

read-only

Remarks

By default, both frame axis values are set to **100**, which means 100 pixels when the field of view is not calibrated, and 100 physical units in case of a calibrated field of view.

EFrameShape.SizeY

Frame Y-axis length.

[VB6]

SizeY As Single

read-only

Remarks

By default, both frame axis values are set to **100**, which means 100 pixels when the field of view is not calibrated, and 100 physical units in case of a calibrated field of view.

EFrameShape.Type

Shape type.

[VB6]

Type As EShapeType

read-only

EGrayscaleDoubleThresholdSegmenter Class

Segments an image using a double threshold on a grayscale image.

Remarks

This segmenter is applicable to [EROIBW8](#) and [EROIBW16](#) grayscale images. It produces coded images with three layers: The Black layer (usually, with index 0) contains the unmasked pixels having a gray value strictly below the low threshold value; the White layer (usually, with index 2) contains the unmasked pixels having a gray value above or equal to the high threshold value; and the Neutral layer (usually, with index 1) contains the remaining unmasked pixels.

Base Class: [EThreeLayersImageSegmenter](#)

PROPERTIES

HighThreshold

Value of the high threshold.

LowThreshold

Value of the low threshold.

G

rayscaleDoubleThresholdSegmenter.HighThreshol
d

Value of the high threshold.

[VB6]

HighThreshold As Long

read-write

EGrayscaleDoubleThresholdSegmenter.LowThresh old

Value of the low threshold.

[VB6]

LowThreshold As Long

read-write

EGrayscaleSingleThresholdSegmenter Class

Segments an image using a single threshold on a grayscale image.

Remarks

This segmenter is applicable to [EROIBW8](#) and [EROIBW16](#) grayscale images. It produces coded images with two layers: the black layer (usually, with index 0) contains the unmasked pixels having a gray value strictly below the threshold value; and the white layer (usually, with index 1) contains the remaining unmasked pixels, i.e. unmasked pixels having a gray value greater or equal to the threshold value. The default thresholding method is minimum residue. If another method is required, the [EGrayscaleSingleThresholdSegmenter::Mode](#) property must be set prior to encoding.

Base Class: [ETwoLayersImageSegmenter](#)

PROPERTIES**AbsoluteThreshold**

Value of the threshold to be used in the case of absolute thresholding.

LastThreshold

Retrieves the actual threshold value that was used for the last image that got encoded by the segmenter.

Mode

Threshold selection mode.

RelativeThreshold

M Fraction of the image pixels that belongs to the black layer in the case of relative thresholding.

E

THODS**IsFirstApplication**

E Checks whether the segmenter has already been used to segment an image.

G

rayscaleSingleThresholdSegmenter.AbsoluteThreshold

Value of the threshold to be used in the case of absolute thresholding.

[VB6]

AbsoluteThreshold As Long

read-write

EGrayscaleSingleThresholdSegmenter::IsFirstApplication

Checks whether the segmenter has already been used to segment an image.

```
[VB6]  
Boolean IsFirstApplication()  
)
```

EGrayscaleSingleThresholdSegmenter::LastThreshold

Retrieves the actual threshold value that was used for the last image that got encoded by the segmenter.

```
[VB6]  
LastThreshold As Long  
read-only
```

Remarks

A call to this method will result in an exception if it is the first time the segmenter is applied. To check whether the segmenter has already been applied, call the [EGrayscaleSingleThresholdSegmenter::IsFirstApplication](#) method.

EGrayscaleSingleThresholdSegmenter.Mode

Threshold selection mode.

[VB6]

Mode As EGrayscaleSingleThreshold

read-write

EGrayscaleSingleThresholdSegmenter.RelativeThreshold

Fraction of the image pixels that belongs to the black layer in the case of relative thresholding.

[VB6]

RelativeThreshold As Single

read-write

EHarrisCornerDetector Class

Manages a complete context for the Harris corner detector.

Remarks

This implementation of the Harris corner detector operates exclusively on a grayscale BW8 images.

PROPERTIES

DerivationScale	Sets the derivation scale.
GradientNormalizationEnabled	Sets whether the gradient is normalized before the computation of the cornerness measure.
IntegrationScale	The integration scale.
SubpixelPrecisionEnabled	Sets whether the sub-pixel interpolation is enabled.
Threshold	Threshold on the cornerness measure for a pixel to be considered as a corner.

M Thresholding mode for the cornerness measure.

E

THODS

Apply	Apply the Harris corner detector on an image/ROI.
EHarrisCornerDetector	Constructs a EHarrisCornerDetector object initialized to its default values.

EHarrisCornerDetector.Apply

Apply the Harris corner detector on an image/ROI.

[VB6]

```
void Apply(  
    EROIBW8 source,  
    EHarrisInterestPoints interestPoints  
)
```

Parameters

source

The source image/ROI.

interestPoints

The container in which to store the interest points.

EHarrisCornerDetector.DerivationScale

Sets the derivation scale.

[VB6]

```
DerivationScale As Single
```

read-write

Remarks

The derivation scale is the standard deviation of the Gaussian Filter used for the noise reduction during the computation of the gradient. Whenever the integration scale is set through [EHarrisCornerDetector::IntegrationScale](#), the derivation scale is reset to its default value, **0.7 * integrationScale**. This is a recommended value, as suggested by the literature.

EHarrisCornerDetector.EHarrisCornerDetector

Constructs a EHarrisCornerDetector object initialized to its default values.

```
[VB6]  
void EHarrisCornerDetector(  
    EHarrisCornerDetector other  
)  
  
void EHarrisCornerDetector(  
)
```

Parameters

other

-

EHarrisCornerDetector.GradientNormalizationEnabled

Sets whether the gradient is normalized before the computation of the cornerness measure.

```
[VB6]  
GradientNormalizationEnabled As Boolean  
read-write
```

Remarks

If this flag is enabled, the values of the X-gradient and of the Y-gradient are first divided by their maximum absolute value (in the internal computations). This results in a cornerness measure that is roughly distributed around the value 1. If this flag is disabled, the cornerness measure will be much greater.

EHarrisCornerDetector.IntegrationScale

The integration scale.

[VB6]

IntegrationScale As Single

read-write

Remarks

The *integration scale* is the standard deviation of the Gaussian filter that is used for scale analysis.

EHarrisCornerDetector.SubpixelPrecisionEnabled

Sets whether the sub-pixel interpolation is enabled.

[VB6]

SubpixelPrecisionEnabled As Boolean

read-write

Remarks

When this flag is enabled, a sub-pixel interpolation is carried on so as to improve the accuracy of the location of the corners, to the expense of a loss of speed.

EHarrisCornerDetector.Threshold

Threshold on the cornerness measure for a pixel to be considered as a corner.

[VB6]

Threshold As Single

read-write

Remarks

If the threshold mode is set to [EHarrisThresholdingMode_Absolute](#), the threshold value is interpreted as an absolute threshold on the cornerness. In this case, the threshold must be a strictly positive real value.

If the threshold mode is set to [EHarrisThresholdingMode_Relative](#), the threshold is expressed as a fraction ranging from 0 to 1 of the maximum value of the cornerness of the source image.

EHarrisCornerDetector.ThresholdingMode

Thresholding mode for the cornerness measure.

[VB6]

ThresholdingMode As EHarrisThresholdingMode

read-write

EHarrisInterestPoints Class

Container class for the results of the Harris corner detector.

Remarks

The [EHarrisCornerDetector](#) class stores its results in this container.

PROPERTIES

PointCount

The number of corner points in the container.

METHODS

<code>Draw</code>	Draws the location of the corner points.
<code>DrawCorner</code>	Draws the location of a specific corner point.
<code>DrawCornerWithCurrentPen</code>	Draws the location of a specific corner point.
<code>DrawWithCurrentPen</code>	Draws the location of the corner points.
<code>EHarrisInterestPoints</code>	Constructs a container for the results of a Harris corner detector.
<code>GetCornerness</code>	Returns the cornerness measure of a corner point.
<code>GetGradientMagnitude</code>	Returns the magnitude of the gradient at a corner point.
<code>GetGradientOrientation</code>	Returns the orientation of the gradient at a corner point.
<code>GetGradientX</code>	Returns the gradient along the X-axis of a corner point.
<code>GetGradientY</code>	Returns the gradient along the Y-axis of a corner point.

GetPoint

Returns the coordinates of a corner point.

GetX

Returns the abscissa of a corner point.

GetY

Returns the ordinate of a corner point.

H

arrisInterestPoints.Draw

Draws the location of the corner points.

[VB6]

```
void Draw(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

void Draw(
    Long graphicContext,
    ERGBColor color,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

void Draw(
    EDrawAdapter graphicContext,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Graphic context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

originX

Horizontal panning factor. By default, no panning occurs.

originY

Vertical panning factor. By default, no panning occurs.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window.

EHarrisInterestPoints.DrawCorner

Draws the location of a specific corner point.

[VB6]

```
void DrawCorner(
    Long graphicContext,
    Long index,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

```
void DrawCorner(
    Long graphicContext,
    ERGBColor color,
    Long index,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

void DrawCorner(
    EDrawAdapter graphicContext,
    Long index,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Graphic context on which to draw.

index

Corner index

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

originX

Horizontal panning factor. By default, no panning occurs.

originY

Vertical panning factor. By default, no panning occurs.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window.

EHarrisInterestPoints.DrawCornerWithCurrentPen

Draws the location of a specific corner point.

[VB6]

```
void DrawCornerWithCurrentPen(
    Long graphicContext,
    Long index,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Graphic context on which to draw.

index

Corner index

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

originX

Horizontal panning factor. By default, no panning occurs.

originY

Vertical panning factor. By default, no panning occurs.

Remarks

Drawing is done in the device context associated to the desired window.

EHarrisInterestPoints.DrawWithCurrentPen

Draws the location of the corner points.

[VB6]

```
void DrawWithCurrentPen(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Graphic context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

*zoomY*Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.*originX*

Horizontal panning factor. By default, no panning occurs.

originY

Vertical panning factor. By default, no panning occurs.

Remarks

Drawing is done in the device context associated to the desired window.

EHarrisInterestPoints.EHarrisInterestPoints

Constructs a container for the results of a Harris corner detector.

[VB6]

```
void EHarrisInterestPoints(
    EHarrisInterestPoints other
)

void EHarrisInterestPoints(
)
```

Parameters

other

-

EHarrisInterestPoints.GetCornerness

Returns the cornerness measure of a corner point.

[VB6]

```
Single GetCornerness(
    Long index
)
```

Parameters

index

The index of the corner point.

EHarrisInterestPoints.GetGradientMagnitude

Returns the magnitude of the gradient at a corner point.

[VB6]

```
Single GetGradientMagnitude(
    Long index
)
```

Parameters

index

The index of the corner point.

EHarrisInterestPoints.GetGradientOrientation

Returns the orientation of the gradient at a corner point.

[VB6]

```
Single GetGradientOrientation(  
    Long index  
)
```

Parameters

index

The index of the corner point.

EHarrisInterestPoints.GetGradientX

Returns the gradient along the X-axis of a corner point.

[VB6]

```
Single GetGradientX(  
    Long index  
)
```

Parameters

index

The index of the corner point.

EHarrisInterestPoints.GetGradientY

Returns the gradient along the Y-axis of a corner point.

[VB6]

```
Single GetGradientY(  
    Long index  
)
```

Parameters

index

The index of the corner point.

EHarrisInterestPoints.GetPoint

Returns the coordinates of a corner point.

[VB6]

```
EPoint GetPoint(  
    Long index  
)
```

Parameters

index

The index of the corner point.

EHarrisInterestPoints.GetX

Returns the abscissa of a corner point.

[VB6]

```
Single GetX(  
    Long index  
)
```

Parameters

index

The index of the corner point.

EHarrisInterestPoints.GetY

Returns the ordinate of a corner point.

```
[VB6]  
Single GetY(  
    Long index  
)
```

Parameters

index

The index of the corner point.

EHarrisInterestPoints.PointCount

The number of corner points in the container.

```
[VB6]  
PointCount As Long  
read-only
```

EHDRCColorFuser Class

A [EHDRCColorFuser](#) instance is a tool that flexibly fuses color images using HDR principles.

METHODS[EHDRColorFuser](#)

Constructors for EHDRColorFuser objects. The image used must be the lightest (slowest shutter speed).

[Fuse](#)

Fuses a dark image with the light image passed during object construction.

[GetFusedImage](#)

ERetrieves the resulting image. Returns false if part of the potential image dynamic is lost due to the chosen output image type.
H

DRColorFuser.EHDRColorFuser

Constructors for EHDRColorFuser objects. The image used must be the lightest (slowest shutter speed).

[VB6]

```
void EHDRColorFuser(
    EROIC24 lightSrc
)

void EHDRColorFuser(
    EROIC48 lightSrc
)
```

Parameters

lightSrc

EHDRColorFuser.Fuse

Fuses a dark image with the light image passed during object construction.

```
[VB6]

void Fuse(  
    EROIC24 darkSrc,  
    Long shutterSpeedFactor  
)  
  
void Fuse(  
    EROIC48 darkSrc,  
    Long shutterSpeedFactor  
)
```

Parameters

darkSrc

Dark input image (higher shutter speed).

shutterSpeedFactor

Shutter speed factor between light and dark image.

EHDRColorFuser.GetFusedImage

Retrieves the resulting image. Returns false if part of the potential image dynamic is lost due to the chosen output image type.

```
[VB6]

Boolean GetFusedImage(  
    EROIC24 dst  
)  
  
Boolean GetFusedImage(  
    EROIC48 dst  
)
```

Parameters*dst*

Output image.

EHDRFuser Class

A [EHDRFuser](#) instance is a tool that flexibly fuses grayscale images using HDR principles.

METHODS[EHDRFuser](#)

Constructors for EHDRFuser objects. The image used must be the lightest (slowest shutter speed).

[Fuse](#)

Fuses a dark image with the light image passed during object construction.

[GetFusedImage](#)

ERetrieves the resulting image. Returns false if part of the potential image dynamic is lost due to the chosen output image type.
H

DRFuser.EHDRFuser

Constructors for EHDRFuser objects. The image used must be the lightest (slowest shutter speed).

```
[VB6]
void EHDRFuser(
    EROIBW8 lightSrc
)
```

```
void EHDRFuser(
    EROIBW16 lightSrc
)
```

Parameters

lightSrc

EHDRFuser.Fuse

Fuses a dark image with the light image passed during object construction.

```
[VB6]

void Fuse(
    EROIBW8 darkSrc,
    Long shutterSpeedFactor
)

void Fuse(
    EROIBW16 darkSrc,
    Long shutterSpeedFactor
)
```

Parameters

darkSrc

Dark input image (higher shutter speed).

shutterSpeedFactor

Shutter speed factor between light and dark image.

EHDRFuser.GetFusedImage

Retrieves the resulting image. Returns false if part of the potential image dynamic is lost due to the chosen output image type.

[VB6]

```
Boolean GetFusedImage(
    EROIBW8 dst
)

Boolean GetFusedImage(
    EROIBW16 dst
)

Boolean GetFusedImage(
    EROIBW32 dst
)
```

Parameters

dst

Output image.

EHitAndMissKernel Class

Class that defines a kernel for the morphological hit-and-miss operations.

PROPERTIES

EndX

Returns the abscissa of the rightmost element of the kernel.

EndY

Returns the abscissa of the bottommost element of the kernel.

StartX

Returns the abscissa of the leftmost element of the kernel.

StartY

M Returns the ordinate of the topmost element of the kernel.

E

THODS**EHitAndMissKernel**

Constructs a EHitAndMissKernel object.

GetValue

Returns the value of an element of the kernel at a given coordinate.

SetSize

Modify the size of the kernel.

SetValue

E Sets the value of an element of the kernel at a given coordinate.

H

itAndMissKernel.EHitAndMissKernel

Constructs a EHitAndMissKernel object.

[VB6]

```
void EHitAndMissKernel(
    EHitAndMissKernel other
)

void EHitAndMissKernel(
    Long startX,
    Long startY,
    Long endX,
    Long endY
)

void EHitAndMissKernel(
    Long halfSizeX,
    Long halfSizeY
)

void EHitAndMissKernel(
)
```

Parameters

other

-

startX

The abscissa of the top leftmost element of the kernel. This value must be less than or equal to zero.

startY

The ordinate of the top leftmost element of the kernel. This value must be less than or equal to zero.

endX

The abscissa of the bottom rightmost element of the kernel. This value must be greater than or equal to zero.

endY

The ordinate of the bottom rightmost element of the kernel. This value must be greater than or equal to zero.

halfSizeX

The half of the kernel width minus 1. This value must be greater than zero.

halfSizeY

The half of the kernel height minus 1. This value must be greater than zero.

Remarks

The constructor without argument creates a centered kernel of size 3x3.

All the elements of the kernel are initialized with [EHitAndMissValue_DontCare](#) values.

If the object is constructed by specifying the halves of the kernel dimensions, the width (resp. the height) of the kernel is given by "2 * halfSizeX + 1" (resp. "2 * halfSizeY + 1"). Otherwise, the width (resp. the height) of the kernel is given by "endX - startX + 1" (resp. "endY - startY + 1").

EHitAndMissKernel.EndX

Returns the abscissa of the rightmost element of the kernel.

[VB6]

EndX As Long

read-only

EHitAndMissKernel.EndY

Returns the abscissa of the bottommost element of the kernel.

[VB6]

EndY As Long

read-only

EHitAndMissKernel.GetValue

Returns the value of an element of the kernel at a given coordinate.

[VB6]

```
EHitAndMissValue GetValue(  
    Long x,  
    Long y  
)
```

Parameters

x

The abscissa of the element.

y

The ordinate of the element.

EHitAndMissKernel.GetSize

Modify the size of the kernel.

[VB6]

```
void SetSize(  
    Long startX,  
    Long startY,  
    Long endX,  
    Long endY  
)  
  
void SetSize(  
    Long halfSizeX,  
    Long halfSizeY  
)
```

Parameters

startX

The abscissa of the top leftmost element of the kernel. This value must be less than or equal to zero.

startY

The ordinate of the top leftmost element of the kernel. This value must be less than or equal to zero.

endX

The abscissa of the bottom rightmost element of the kernel. This value must be greater than or equal to zero.

endY

The ordinate of the bottom rightmost element of the kernel. This value must be greater than or equal to zero.

halfSizeX

The half of the kernel width minus 1. This value must be greater than zero.

halfSizeY

The half of the kernel height minus 1. This value must be greater than zero.

Remarks

All the elements of the kernel are initialized with [EHitAndMissValue_DontCare](#) values.

If the object is constructed by specifying the halves of the kernel dimensions, the width (resp. the height) of the kernel is given by "2 * halfSizeX + 1" (resp. "2 * halfSizeY + 1"). Otherwise, the width (resp. the height) of the kernel is given by "endX - startX + 1" (resp. "endY - startY + 1").

EHitAndMissKernel.SetValue

Sets the value of an element of the kernel at a given coordinate.

[VB6]

```
void SetValue(
    Long x,
    Long y,
    EHitAndMissValue value
)
```

Parameters

x

The abscissa of the element.

y

The ordinate of the element.

value

The value of the element.

EHitAndMissKernel.StartX

Returns the abscissa of the leftmost element of the kernel.

[VB6]

StartX As Long

read-only

EHitAndMissKernel.StartY

Returns the ordinate of the topmost element of the kernel.

[VB6]

StartY As Long

read-only

EHole Class

This class represents a hole inside an object (blob) of an encoded image.

Remarks

This class inherits from the ECodedElement class and provides an additional method to retrieve the parent object of a particular hole.

Base Class: [ECodedElement](#)

PROPERTIES

ParentObjectIndex

Returns the index of the parent object of the hole.

H

ole.ParentObjectIndex

Returns the index of the parent object of the hole.

[VB6]

ParentObjectIndex As Long

read-only

EImageBW1 Class

The EImageBW1 class is used to represent rectangular regions of interest inside **EBW1** black and white images. See ROIs.

Base Class: [EROIBW1](#)

METHODS

EImageBW1

Constructs a EImageBW1 image.

GetBitIndex

-

operator=

Copies a EImageBW1 image.

ElImageBW1.ElImageBW1

Constructs a ElImageBW1 image.

```
[VB6]  
void ElImageBW1()  
)  
  
void ElImageBW1(  
Long width,  
Long height  
)  
  
void ElImageBW1(  
ElImageBW1 other  
)
```

Parameters

width

The width, in pixels.

height

The height, in pixels.

other

Another ElImageBW1 object.

Remarks

The constructor with no arguments creates a zero-sized image. You can modify the image size by calling [EBaseROI::SetSize](#) method. The sizing constructor constructs an image of the given size. See [EBaseROI::SetSize](#) for informations about the sizing restrictions. The copy constructor copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

ElImageBW1.GetBitIndex

[VB6]

```
Unsupported variant type GetBitIndex(  
    Long x,  
    Long y  
)
```

Parameters

x

-

y

-

EImageBW1.operator=

Copies a EImageBW1 image.

[VB6]

```
EImageBW1 operator=(  
    EImageBW1 other  
)
```

Parameters

other

Another EImageBW1 object.

Remarks

This method copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

ElImageBW16 Class

The ElImageBW16 class is used to represent rectangular regions of interest inside [EBW16](#) gray-level images. See ROIs.

Base Class: [EROIBW16](#)

METHODS

[ElImageBW16](#)

Constructs a ElImageBW16 image.

[operator=](#)

Copies a ElImageBW16 image.

[ImageBW16](#).ElImageBW16

Constructs a ElImageBW16 image.

[VB6]

```
void ElImageBW16()
)

void ElImageBW16(
    Long width,
    Long height
)

void ElImageBW16(
    ElImageBW16 other
)
```

Parameters

width

The width, in pixels.

height

The height, in pixels.

other

Another EImageBW16 object.

Remarks

The constructor with no arguments creates a zero-sized image. You can modify the image size by calling [EBaseROI::SetSize](#) method. The sizing constructor constructs an image of the given size. See [EBaseROI::SetSize](#) for informations about the sizing restrictions. The copy constructor copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

EImageBW16.operator=

Copies a EImageBW16 image.

[VB6]

```
EImageBW16 operator=(  
    EImageBW16 other  
)
```

Parameters

other

Another EImageBW16 object.

Remarks

This method copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

EImageBW32 Class

The EImageBW32 class is used to represent rectangular regions of interest inside [EBW32](#) gray-level images. See ROIs.

Base Class: [EROIBW32](#)

METHODS

`EImageBW32`

Constructs a `EImageBW32` image.

`operator=`

Copies a `EImageBW32` image.

`imageBW32.EImageBW32`

Constructs a `EImageBW32` image.

[VB6]

```
void EImageBW32 ()  
void EImageBW32 ( Long width, Long height )  
void EImageBW32 ( EImageBW32 other )
```

Parameters

`width`

The width, in pixels.

`height`

The height, in pixels.

`other`

Another `EImageBW32` object.

Remarks

The constructor with no arguments creates a zero-sized image. You can modify the image size by calling [EBaseROI::SetSize](#) method. The sizing constructor constructs an image of the given size. See [EBaseROI::SetSize](#) for informations about the sizing restrictions. The copy constructor copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

EImageBW32.operator=

Copies a EImageBW32 image.

[VB6]

```
EImageBW32 operator=
EImageBW32 other
)
```

Parameters

other

Another EImageBW32 object.

Remarks

This method copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

EImageBW8 Class

The EImageBW8 class is used to represent rectangular regions of interest inside [EBW8](#) gray-level images. See ROIs.

Base Class: [EROIBW8](#)

METHODS

[EImageBW8](#)

	Constructs a EImageBW8 image.
operator=	Copies a EImageBW8 image.

mageBW8.EImageBW8

Constructs a EImageBW8 image.

```
[VB6]  
void EImageBW8()  
)  
  
void EImageBW8(  
Long width,  
Long height  
)  
  
void EImageBW8(  
EImageBW8 other  
)
```

Parameters

width

The width, in pixels.

height

The height, in pixels.

other

Another EImageBW8 object.

Remarks

The constructor with no arguments creates a zero-sized image. You can modify the image size by calling [EBaseROI::SetSize](#) method. The sizing constructor constructs an image of the given size. See [EBaseROI::SetSize](#) for informations about the sizing restrictions. The copy constructor copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

EImageBW8.operator=

Copies a EImageBW8 image.

[VB6]

```
EImageBW8 operator=(  
    EImageBW8 other  
)
```

Parameters

other

Another EImageBW8 object.

Remarks

This method copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

EImageC15 Class

The EImageC15 class is used to represent rectangular regions of interest inside EC15 color images. See ROIs.

Base Class: EROI15

METHODS

[EImageC15](#)

Constructs a EImageC15 image.

[operator=](#)

Copies a EImageC15 image.

ElImageC15.ElImageC15

Constructs a ElImageC15 image.

```
[VB6]  
void ElImageC15()  
)  
  
void ElImageC15(  
Long width,  
Long height  
)  
  
void ElImageC15(  
ElImageC15 other  
)
```

Parameters

width

The width, in pixels.

height

The height, in pixels.

other

Another ElImageC15 object.

Remarks

The constructor with no arguments creates a zero-sized image. You can modify the image size by calling [EBaseROI::SetSize](#) method. The sizing constructor constructs an image of the given size. See [EBaseROI::SetSize](#) for informations about the sizing restrictions. The copy constructor copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

ElImageC15.operator=

Copies a ElImageC15 image.

[VB6]

```
EImageC15 operator=(
    EImageC15 other
)
```

Parameters

other

Another EImageC15 object.

Remarks

This method copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

EImageC16 Class

The EImageC16 class is used to represent rectangular regions of interest inside EC16 color images. See ROIs.

Base Class: EROI^C16

METHODS

[EImageC16](#)

Constructs a EImageC16 image.

[operator=](#)

Copies a EImageC16 image.

mageC16.EImageC16

Constructs a EImageC16 image.

```
[VB6]
void EImageC16(
)
void EImageC16(
    Long width,
    Long height
)
void EImageC16(
    EImageC16 other
)
```

Parameters

width

The width, in pixels.

height

The height, in pixels.

other

Another EImageC16 object.

Remarks

The constructor with no arguments creates a zero-sized image. You can modify the image size by calling [EBaseROI::SetSize](#) method. The sizing constructor constructs an image of the given size. See [EBaseROI::SetSize](#) for informations about the sizing restrictions. The copy constructor copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

EImageC16.operator=

Copies a EImageC16 image.

```
[VB6]
EImageC16 operator=(
    EImageC16 other
)
```

Parameters

other

Another ElImageC16 object.

Remarks

This method copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

ElImageC24 Class

The ElImageC24 class is used to represent rectangular regions of interest inside [EC24](#) color images. See ROIs.

Base Class: [EROIC24](#)

METHODS

[ElImageC24](#)

Constructs a ElImageC24 image.

[operator=](#)

Copies a ElImageC24 image.

[ElImageC24](#)

Constructs a ElImageC24 image.

[VB6]

```
void ElImageC24(  
)
```

```

void EImageC24(
    Long width,
    Long height
)

void EImageC24(
    EImageC24 other
)

```

Parameters*width*

The width, in pixels.

height

The height, in pixels.

other

Another EImageC24 object.

Remarks

The constructor with no arguments creates a zero-sized image. You can modify the image size by calling [EBaseROI::SetSize](#) method. The sizing constructor constructs an image of the given size. See [EBaseROI::SetSize](#) for informations about the sizing restrictions. The copy constructor copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

EImageC24.operator=

Copies a EImageC24 image.

[VB6]

```

EImageC24 operator=(
    EImageC24 other
)

```

Parameters*other*

Another EImageC24 object.

Remarks

This method copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

EImageC24A Class

The EImageC24A class is used to represent rectangular regions of interest inside [EC24A](#) color images. See ROIs.

Base Class: [EROIC24A](#)

METHODS

[EImageC24A](#)

Constructs a EImageC24A image.

[operator=](#)

Copies a EImageC24A image.

[imageC24A.EImageC24A](#)

Constructs a EImageC24A image.

```
[VB6]  
void EImageC24A()  
)  
  
void EImageC24A(  
Long width,  
Long height  
)
```

```
void EImageC24A(
    EImageC24A other
)
```

Parameters*width*

The width, in pixels.

height

The height, in pixels.

other

Another EImageC24A object.

Remarks

The constructor with no arguments creates a zero-sized image. You can modify the image size by calling [EBaseROI::SetSize](#) method. The sizing constructor constructs an image of the given size. See [EBaseROI::SetSize](#) for informations about the sizing restrictions. The copy constructor copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

EImageC24A.operator=

Copies a EImageC24A image.

[VB6]

```
EImageC24A operator=
    EImageC24A other
)
```

Parameters*other*

Another EImageC24A object.

Remarks

This method copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

ElImageC48 Class

The ElImageC48 class is used to represent rectangular regions of interest inside EC48 color images. See ROIs.

Base Class: EROI48

METHODS

ElImageC48

Constructs a ElImageC48 image.

operator=

Copies a ElImageC48 image.

mageC48.ElImageC48

Constructs a ElImageC48 image.

[VB6]

```
void ElImageC48()
)

void ElImageC48(
    Long width,
    Long height
)

void ElImageC48(
    ElImageC48 other
)
```

Parameters

width

The width, in pixels.

height

The height, in pixels.

other

Another EImageC48 object.

Remarks

The constructor with no arguments creates a zero-sized image. You can modify the image size by calling [EBaseROI::SetSize](#) method. The sizing constructor constructs an image of the given size. See [EBaseROI::SetSize](#) for informations about the sizing restrictions. The copy constructor copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

EImageC48.operator=

Copies a EImageC48 image.

[VB6]

```
EImageC48 operator=
EImageC48 other
)
```

Parameters

other

Another EImageC48 object.

Remarks

This method copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

EImageEncoder Class

This class is responsible for the encoding of an image into an [ECodedImage2](#) object.

Remarks

This class is responsible for the extraction of the runs in a source image, the aggregation of the runs into objects, as well as the proper handling of the continuous mode. It also provides methods to configure the image segmentation process.

The segmentation process classifies the pixels of the source image according to their value to create a set of layers. In each layer taken separately, the encoding process then assembles the connected pixels to build the coded elements (blobs).

By default, the segmentation method consists of grayscale single thresholding, with automatic threshold selection (as determined by the minimum residue rule).

PROPERTIES

`BinaryImageSegmenter`

Returns a reference to the internal instance of the the segmenter for binary images, in order to set its parameters.

`ColorRangeThresholdSegmenter`

Returns a reference to the internal instance of the the segmenter for color-range thresholding, in order to set its parameters.

`ColorSingleThresholdSegmenter`

Returns a reference to the internal instance of the the segmenter for single color thresholding, in order to set its parameters.

`ContinuousModeEnabled`

Continuous mode enabling status.

`ContinuousModeMaxHeight`

Maximum number of rows that are kept in memory in the continuous mode.

`EncodingConnexity`

Connexity mode.

GrayscaleDoubleThresholdSegmenter	Returns a reference to the internal instance of the segmenter for double color thresholding, in order to set its parameters.
GrayscaleSingleThresholdSegmenter	Returns a reference to the internal instance of the segmenter for single grayscale thresholding, in order to set its parameters.
ImageRangeSegmenter	Returns a reference to the internal instance of the segmenter for pixel-by-pixel, range-based thresholding, in order to set its parameters.
LabeledImageSegmenter	Returns a reference to the internal instance of the segmenter that maps the value of the pixels directly to a layer index, in order to set its parameters.
ReferenceImageSegmenter	Returns a reference to the internal instance of the segmenter for single pixel-by-pixel thresholding, in order to set its parameters.
SegmentationMethod	M Segmentation method used during the encoding.
THODS	E
EImageEncoder	Constructs an image encoder.
Encode	Encodes an image or an ROI as a coded image.

[FlushContinuousMode](#)

Resets the continuous mode, emptying the internal memory, after writing the objects that are not yet completed in a coded image.

[ResetContinuousMode](#)

ElImageEncoder.BinaryImage

Resets the continuous mode, emptying the internal memory.

S

egmenter

Returns a reference to the internal instance of the the segmenter for binary images, in order to set its parameters.

[VB6]

BinaryImageSegmenter As EBinaryImageSegmenter

read-only

ElImageEncoder.ColorRangeThresholdSegmenter

Returns a reference to the internal instance of the the segmenter for color-range thresholding, in order to set its parameters.

[VB6]

```
ColorRangeThresholdSegmenter As EColorRangeThresholdSegmenter
```

read-only

ElImageEncoder.ColorSingleThresholdSegmenter

Returns a reference to the internal instance of the the segmenter for single color thresholding, in order to set its parameters.

[VB6]

```
ColorSingleThresholdSegmenter As EColorSingleThresholdSegmenter
```

read-only

ElImageEncoder.ContinuousModeEnabled

Continuous mode enabling status.

[VB6]

```
ContinuousModeEnabled As Boolean
```

read-write

Remarks

In the continuous mode, objects are constructed over a sequence of images: The image encoder encodes only the objects that contain no run touching the last row of the source image. The objects touching the inferior border of the image are not written in the coded image: These objects are indeed expected to continue in the subsequent image chunks. Such objects are kept in memory, and are consumed when analyzing the subsequent images.

ElImageEncoder.ContinuousModeMaxHeight

Maximum number of rows that are kept in memory in the continuous mode.

[VB6]

ContinuousModeMaxHeight As Long

read-write

Remarks

This property can be used to put a bound on the size of the internal memory of the image encoder in the continuous mode. If this property is set to zero, then memory can grow arbitrarily (there is no maximum number of rows).

ElImageEncoder.ElImageEncoder

Constructs an image encoder.

[VB6]

```
void ElImageEncoder(
    ElImageEncoder other
)
void ElImageEncoder(
)
```

Parameters

other

ElImageEncoder.Encode

Encodes an image or an ROI as a coded image.

[VB6]

```
void Encode(
    EROIBW1 sourceImage,
    ECodedImage2 codedImage
)

void Encode(
    EROIBW8 sourceImage,
    ECodedImage2 codedImage
)

void Encode(
    EROIBW16 sourceImage,
    ECodedImage2 codedImage
)

void Encode(
    EROIC24 sourceImage,
    ECodedImage2 codedImage
)

void Encode(
    EROIBW1 sourceImage,
    EROIBW8 inputMask,
    ECodedImage2 codedImage
)

void Encode(
    EROIBW8 sourceImage,
    EROIBW8 inputMask,
    ECodedImage2 codedImage
)

void Encode(
    EROIBW16 sourceImage,
    EROIBW8 inputMask,
    ECodedImage2 codedImage
)
```

```
void Encode(
    EROIC24 sourceImage,
    EROIBW8 inputMask,
    ECodedImage2 codedImage
)
```

Parameters*sourceImage*

The input image that is to be encoded.

codedImage

The coded image that will hold the result of the encoding process.

inputMask

The possible input Flexible Mask that restricts the encoding. The input mask is a grayscale image having the same height and the same width as the source image. Any pixel in the source image that is covered by a value of **0** in the input mask will not get encoded in any layer. Any other pixel value in the input mask causes the pixel to be a candidate for the encoding.

Remarks

The previous content of the result coded image is discarded.

ElImageEncoder.EncodingConnexity

Connexity mode.

[VB6]**EncodingConnexity As EEncodingConnexity**

read-write

Remarks

The connexity mode specifies the conditions that must hold for neighboring pixels to belong to the same object.

ElImageEncoder.FlushContinuousMode

Resets the continuous mode, emptying the internal memory, after writing the objects that are not yet completed in a coded image.

[VB6]

```
void FlushContinuousMode(  
    ECodedImage2 codedImage  
)
```

Parameters

codedImage

The coded image in which the not-yet-completed objects are written.

ElImageEncoder.GrayscaleDoubleThresholdSegmenter

Returns a reference to the internal instance of the the segmenter for double color thresholding, in order to set its parameters.

[VB6]

```
GrayscaleDoubleThresholdSegmenter As  
EGrayscaleDoubleThresholdSegmenter  
read-only
```

ElImageEncoder.GrayscaleSingleThresholdSegmenter

Returns a reference to the internal instance of the the segmenter for single grayscale thresholding, in order to set its parameters.

[VB6]

```
GrayscaleSingleThresholdSegmenter As  
EGrayscaleSingleThresholdSegmenter
```

read-only

ElImageEncoder.ImageRangeSegmenter

Returns a reference to the internal instance of the the segmenter for pixel-by-pixel, range-based thresholding, in order to set its parameters.

[VB6]

```
ImageRangeSegmenter As EImageRangeSegmenter
```

read-only

ElImageEncoder.LabeledImageSegmenter

Returns a reference to the internal instance of the the segmenter thats maps the value of the pixels directly to a layer index, in order to set its parameters.

[VB6]

```
LabeledImageSegmenter As ELabeledImageSegmenter
```

read-only

ElImageEncoder.ReferenceImageSegmenter

Returns a reference to the internal instance of the the segmenter for single pixel-by-pixel thresholding, in order to set its parameters.

[VB6]

```
ReferenceImageSegmenter As EReferenceImageSegmenter
```

read-only

ElImageEncoder.ResetContinuousMode

Resets the continuous mode, emptying the internal memory.

[VB6]

```
void ResetContinuousMode()  
)
```

ElImageEncoder.SegmentationMethod

Segmentation method used during the encoding.

[VB6]

SegmentationMethod As ESegmentationMethod

read-write

ElImageRangeSegmenter Class

Segments an image using a pixel-by-pixel double threshold given as two images.

Remarks

This segmenter is applicable to [EROIBW8](#), [EROIBW16](#) and [EROIC24](#) images. It produces coded images with two layers. The low threshold and the high threshold are defined for each pixel individually by means of two reference images of the same type as the source image: the Low Image and the High Image.

For grayscales images, the White layer (usually, with index 1) contains unmasked pixels having a gray value in a range defined by the gray value of the corresponding unmasked pixels in the Low Image and the High Image.

For RGB color images, the White layer (usually, with index 1) contains unmasked pixels having a color inside the cube of the color space defined by the colors of the corresponding unmasked pixels in the Low Image and the High Image.

The Black layer (usually, with index 0) contains the remaining unmasked pixels.

Base Class: [ETwoLayersImageSegmenter](#)

PROPERTIES

[BlackLayerEncoded](#)

Black layer encoding status.

[BlackLayerIndex](#)

Index of the black layer in the destination coded image.

[HighImageBW16](#)

High image for the segmentation of [EROIBW16](#) images.

HighImageBW8	High image for the segmentation of EROIBW8 images.
HighImageC24	High image for the segmentation of EROIC24 images.
LowImageBW16	Low image for the segmentation of EROIBW16 images.
LowImageBW8	Low image for the segmentation of EROIBW8 images.
LowImageC24	Low image for the segmentation of EROIC24 images.
WhiteLayerEncoded	White layer encoding status.
WhiteLayerIndex	Index of the white layer in the destination coded image.

`imageRangeSegmenter.BlackLayerEncoded`

Black layer encoding status.

[VB6]

`BlackLayerEncoded As Boolean`

read-write

ElImageRangeSegmenter.BlackLayerIndex

Index of the black layer in the destination coded image.

[VB6]

BlackLayerIndex As Long

read-write

Remarks

Setting this property automatically switches on the encoding of the black layer.

ElImageRangeSegmenter.HighImageBW16

High image for the segmentation of [EROIBW16](#) images.

[VB6]

HighImageBW16 As EROIBW16

read-write

ElImageRangeSegmenter.HighImageBW8

High image for the segmentation of [EROIBW8](#) images.

[VB6]

HighImageBW8 As EROIBW8

read-write

ElImageRangeSegmenter.HighImageC24

High image for the segmentation of [EROIC24](#) images.

[VB6]

HighImageC24 As EROIC24

read-write

ElImageRangeSegmenter.LowImageBW16

Low image for the segmentation of [EROIBW16](#) images.

[VB6]

LowImageBW16 As EROIBW16

read-write

ElImageRangeSegmenter.LowImageBW8

Low image for the segmentation of [EROIBW8](#) images.

[VB6]

LowImageBW8 As EROIBW8

read-write

ElImageRangeSegmenter.LowImageC24

Low image for the segmentation of [EROIC24](#) images.

[VB6]

LowImageC24 As EROIC24

read-write

ElImageRangeSegmenter.WhiteLayerEncoded

White layer encoding status.

[VB6]

WhiteLayerEncoded As Boolean

read-write

ElImageRangeSegmenter.WhiteLayerIndex

Index of the white layer in the destination coded image.

[VB6]

WhiteLayerIndex As Long

read-write

Remarks

Setting this property automatically switches on the encoding of the white layer.

EImageSegmenter Class

Base class from which all the segmenters derive.

Derived Class(es): [ETwoLayersImageSegmenter](#) [EThreeLayersImageSegmenter](#)
[ELabeledImageSegmenter](#)

EKernel Class

Kernel for use in convolution operations.

PROPERTIES

Gain

Global gain.

Offset

Global offset (a constant added to the convolution result).

OutsideValue

Out-of-limits image value (only influences the result along image edges).

RawDataPtr	Pointer to the upper left convolution coefficient.
Rectifier	Rectification mode. This property allows specifying how negative convolution result values are handled.
SizeX	Number of coefficients along a row.
SizeY	M Number of coefficients along a column. E
THODS	
EKernel	Constructs an EKernel object.
GetKernelData	Returns the convolution coefficient of given indices.
SetKernelData	Sets the convolution coefficient values at the given indices.
SetSize	-

EKernel.EKernel

Constructs an EKernel object.

```
[VB6]  
void EKernel(  
    EKernel other  
)  
  
void EKernel(  
)  
  
void EKernel(  
    Integer sizeX,  
    Integer sizeY,  
    Single gain,  
    Long offset,  
    EKernelRectifier rectifier,  
    Long outsideValue  
)  
  
void EKernel(  
    EKernelType KernelType  
)
```

Parameters

other

-

sizeX

Number of coefficients along a row.

sizeY

Number of coefficients along a column.

gain

Global gain.

offset

Global offset.

rectifier

Rectification mode, as defined by [EKernelRectifier](#).

outsideValue

Out-of-limits image value.

*KernelType*Kernel type, as defined by [EKernelType](#).**Remarks**

The default constructor constructs a void kernel. A void kernel has no associated convolution coefficients. The sizing constructor constructs a kernel of given size and global parameters. The third constructor constructs a kernel of a predefined type.

EKernel.Gain

Global gain.

[VB6]

Gain As Single

read-write

Remarks

Before the global gain is applied, the coefficients are normalized so that their sum equals one, unless their sum equals zero (as is the case for a derivation operator). The rectification enables to handle the negative values that may appear after convolution.

EKernel.GetKernelData

Returns the convolution coefficient of given indices.

[VB6]

```
void GetKernelData(
    Long columnIndex,
    Long rowIndex,
    Single coefficientValue
)
```

Parameters*columnIndex*

Column index, from **0** on, increasing rightwards.

rowIndex

Row index, from **0** on, increasing downwards.

coefficientValue

Reference to the coefficient value.

EKernel.Offset

Global offset (a constant added to the convolution result).

[VB6]

Offset As Long

read-write

EKernel.OutsideValue

Out-of-limits image value (only influences the result along image edges).

[VB6]

OutsideValue As Long

read-write

EKernel.RawDataPtr

Pointer to the upper left convolution coefficient.

[VB6]

RawDataPtr As Long

read-only

Remarks

This pointer is actually the base address of a float array containing all coefficients.

EKernel.Rectifier

Rectification mode. This property allows specifying how negative convolution result values are handled.

[VB6]

Rectifier As EKernelRectifier

read-write

EKernel.SetKernelData

Sets the convolution coefficient values at the given indices.

[VB6]

```
void SetKernelData(
    Long columnIndex,
    Long rowIndex,
    Single coefficientValue
)
```

```
void SetKernelData(
    Single coefficientValue00,
    Single coefficientValue10,
    Single coefficientValue20,
    Single coefficientValue01,
    Single coefficientValue11,
    Single coefficientValue21,
    Single coefficientValue02,
    Single coefficientValue12,
    Single coefficientValue22
)

void SetKernelData(
    Single coefficientValue00,
    Single coefficientValue10,
    Single coefficientValue20,
    Single coefficientValue30,
    Single coefficientValue40,
    Single coefficientValue01,
    Single coefficientValue11,
    Single coefficientValue21,
    Single coefficientValue31,
    Single coefficientValue41,
    Single coefficientValue02,
    Single coefficientValue12,
    Single coefficientValue22,
    Single coefficientValue32,
    Single coefficientValue42,
    Single coefficientValue03,
    Single coefficientValue13,
    Single coefficientValue23,
    Single coefficientValue33,
    Single coefficientValue43,
    Single coefficientValue04,
    Single coefficientValue14,
    Single coefficientValue24,
    Single coefficientValue34,
    Single coefficientValue44
)
```

```
void SetKernelData(
    Single coefficientValue00,
    Single coefficientValue10,
    Single coefficientValue20,
    Single coefficientValue30,
    Single coefficientValue40,
    Single coefficientValue50,
    Single coefficientValue60,
    Single coefficientValue01,
    Single coefficientValue11,
    Single coefficientValue21,
    Single coefficientValue31,
    Single coefficientValue41,
    Single coefficientValue51,
    Single coefficientValue61,
    Single coefficientValue02,
    Single coefficientValue12,
    Single coefficientValue22,
    Single coefficientValue32,
    Single coefficientValue42,
    Single coefficientValue52,
    Single coefficientValue62,
    Single coefficientValue03,
    Single coefficientValue13,
    Single coefficientValue23,
    Single coefficientValue33,
    Single coefficientValue43,
    Single coefficientValue53,
    Single coefficientValue63,
    Single coefficientValue04,
    Single coefficientValue14,
    Single coefficientValue24,
    Single coefficientValue34,
    Single coefficientValue44,
    Single coefficientValue54,
    Single coefficientValue64,
    Single coefficientValue05,
    Single coefficientValue15,
    Single coefficientValue25,
    Single coefficientValue35,
    Single coefficientValue45,
    Single coefficientValue55,
    Single coefficientValue65,
    Single coefficientValue06,
    Single coefficientValue16,
    Single coefficientValue26,
```

```
Single coefficientValue36,  
Single coefficientValue46,  
Single coefficientValue56,  
Single coefficientValue66  
)
```

Parameters

columnIndex

Column index, from **0** on, increasing rightwards.

RowIndex

Row index, from **0** on, increasing downwards.

coefficientValue

New coefficientValue.

coefficientValue00

Coefficient value at corresponding column and row indices.

coefficientValue10

Coefficient value at corresponding column and row indices.

coefficientValue20

Coefficient value at corresponding column and row indices.

coefficientValue01

Coefficient value at corresponding column and row indices.

coefficientValue11

Coefficient value at corresponding column and row indices.

coefficientValue21

Coefficient value at corresponding column and row indices.

coefficientValue02

Coefficient value at corresponding column and row indices.

coefficientValue12

Coefficient value at corresponding column and row indices.

coefficientValue22

Coefficient value at corresponding column and row indices.

coefficientValue30

Coefficient value at corresponding column and row indices.

coefficientValue40

Coefficient value at corresponding column and row indices.

coefficientValue31

Coefficient value at corresponding column and row indices.

coefficientValue41
Coefficient value at corresponding column and row indices.
coefficientValue32
Coefficient value at corresponding column and row indices.
coefficientValue42
Coefficient value at corresponding column and row indices.
coefficientValue03
Coefficient value at corresponding column and row indices.
coefficientValue13
Coefficient value at corresponding column and row indices.
coefficientValue23
Coefficient value at corresponding column and row indices.
coefficientValue33
Coefficient value at corresponding column and row indices.
coefficientValue43
Coefficient value at corresponding column and row indices.
coefficientValue04
Coefficient value at corresponding column and row indices.
coefficientValue14
Coefficient value at corresponding column and row indices.
coefficientValue24
Coefficient value at corresponding column and row indices.
coefficientValue34
Coefficient value at corresponding column and row indices.
coefficientValue44
Coefficient value at corresponding column and row indices.
coefficientValue50
Coefficient value at corresponding column and row indices.
coefficientValue60
Coefficient value at corresponding column and row indices.
coefficientValue51
Coefficient value at corresponding column and row indices.
coefficientValue61
Coefficient value at corresponding column and row indices.
coefficientValue52
Coefficient value at corresponding column and row indices.
coefficientValue62
Coefficient value at corresponding column and row indices.
coefficientValue53

Coefficient value at corresponding column and row indices.
coefficientValue63

Coefficient value at corresponding column and row indices.
coefficientValue54

Coefficient value at corresponding column and row indices.
coefficientValue64

Coefficient value at corresponding column and row indices.
coefficientValue05

Coefficient value at corresponding column and row indices.
coefficientValue15

Coefficient value at corresponding column and row indices.
coefficientValue25

Coefficient value at corresponding column and row indices.
coefficientValue35

Coefficient value at corresponding column and row indices.
coefficientValue45

Coefficient value at corresponding column and row indices.
coefficientValue55

Coefficient value at corresponding column and row indices.
coefficientValue65

Coefficient value at corresponding column and row indices.
coefficientValue06

Coefficient value at corresponding column and row indices.
coefficientValue16

Coefficient value at corresponding column and row indices.
coefficientValue26

Coefficient value at corresponding column and row indices.
coefficientValue36

Coefficient value at corresponding column and row indices.
coefficientValue46

Coefficient value at corresponding column and row indices.
coefficientValue56

Coefficient value at corresponding column and row indices.
coefficientValue66

Coefficient value at corresponding column and row indices.

Remarks

The function can also set the coefficient values for 3x3, 5x5 and 7x7 kernels.

EKernel.GetSize

-

[VB6]

```
void SetSize(  
    Integer n16SizeX,  
    Integer n16SizeY  
)
```

Parameters

n16SizeX

n16SizeY

EKernel.SizeX

Number of coefficients along a row.

[VB6]

SizeX As Integer

read-only

EKernel.SizeY

Number of coefficients along a column.

[VB6]

SizeY As Integer

read-only

ELabeledImageSegmenter Class

Segments an image by mapping the value of the pixels directly to a layer index.

Remarks

This segmenter is applicable to [EROIBW8](#) and [EROIBW16](#) grayscale images. It produces coded images with a varying number of layers. The layer with index **N** contains all the unmasked pixels having a gray value equal to **N**.

By default, the segmentation is restricted to the range of layers whose index is between **0** and **255** (inclusive). This default range can be changed through [ELabeledImageSegmenter::MinLayer](#) and [ELabeledImageSegmenter::MaxLayer](#).

Base Class: [EImageSegmenter](#)

PROPERTIES

[MaxLayer](#)

High index of the range of layers to be encoded.

[MinLayer](#)

Low index of the range of layers to be encoded.

`ELabeledImageSegmenter.MaxLayer`

High index of the range of layers to be encoded.

[VB6]

MaxLayer As EBW16

read-write

ELabeledImageSegmenter.MinLayer

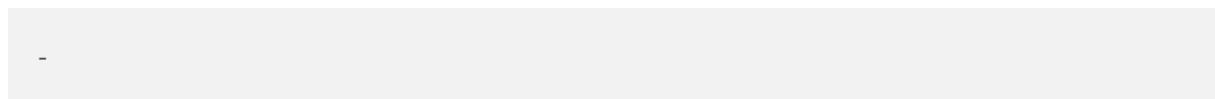
Low index of the range of layers to be encoded.

[VB6]

MinLayer As EBW16

read-write

ELandmark Class



PROPERTIES

SensorX



SensorY



WorldX



WorldY



METHODS

ELandmark

operator=

andmark.ELandmark

[VB6]

```
void ELandmark(  
    ELandmark other  
)  
  
void ELandmark(  
)
```

Parameters

other

ELandmark.operator=

[VB6]

```
ELandmark operator=(  
    ELandmark other  
)
```

Parameters

other

ELandmark.SensorX

-

[VB6]
SensorX As Single
read-write

ELandmark.SensorY

-

[VB6]
SensorY As Single
read-write

ELandmark.WorldX

-

[VB6]

WorldX As Single

read-write

ELandmark.WorldY

-

[VB6]

WorldY As Single

read-write

ELaserLineExtractor Class

Manages a laser line extraction context.

PROPERTIES

AnalysisMode

Analysis mode.

AnalysisThreshold

Analysis threshold. Set this value to eliminate noise.

DepthMap

Returns the current depth map.

EnableSmoothing

Enables or disables grayscale profile smoothing before extraction.

Profile

MReturns the last extracted profile.

THODS**ELaserLineExtractor**

Creates an **ELaserLineExtractor** object.

ExtractProfileFromFrame

Extracts a profile from a frame and adds it to the current depth map. Returns true if the depth map is complete and ready for further processing.

SetSmoothingParameters

ESets the parameters of the smoothing kernel.

L

aserLineExtractor.AnalysisMode

Analysis mode.

[VB6]

AnalysisMode As EMaximumAnalysisMode

read-write

ELaserLineExtractor.AnalysisThreshold

Analysis threshold. Set this value to eliminate noise.

[VB6]

AnalysisThreshold As Long

read-write

Remarks

In the center of gravity (COG) analysis mode, this threshold is used to discriminate peaks and should be set accordingly.

ELaserLineExtractor.DepthMap

Returns the current depth map.

[VB6]

DepthMap As EDepthMap16

read-only

Remarks

Should be called only when the previous call to ExtractProfileFromFrame() returned true. Otherwise, the depth map returns will be incomplete.

ELaserLineExtractor.ELaserLineExtractor

Creates an [ELaserLineExtractor](#) object.

[VB6]

```
void ELaserLineExtractor(  
    Long frameWidth,  
    Long frameHeight,  
    Long numFramesPerMap  
)
```

Parameters

frameWidth

Width of the frames from which the profiles will be extracted.

frameHeight

Height of the frames from which the profiles will be extracted.

numFramesPerMap

Number of frames (and thus profiles) to be used per depth map.

ELaserLineExtractor.EnableSmoothing

Enables or disables grayscale profile smoothing before extraction.

[VB6]

EnableSmoothing As Boolean

read-write

ELaserLineExtractor.ExtractProfileFromFrame

Extracts a profile from a frame and adds it to the current depth map. Returns true if the depth map is complete and ready for further processing.

[VB6]

```
Boolean ExtractProfileFromFrame(
    EROIBW8 frame
)
```

Parameters

frame

Frame from which the profile will be extracted.

ELaserLineExtractor.Profile

Returns the last extracted profile.

[VB6]

```
Profile As ()Single  
read-only
```

Remarks

If a point could not be extracted, its value will be set to FLOAT_MAX.

ELaserLineExtractor.SetSmoothingParameters

Sets the parameters of the smoothing kernel.

[VB6]

```
void SetSmoothingParameters (
    Long param0,
    Long param1,
    Long param2
)
```

Parameters*param0*

First kernel parameter.

param1

Second kernel parameter.

param2

Third kernel parameter.

Remarks

If enabled, the smoothing will be performed using the following formula: $f[i] = (f[i-1] * param0) + (f[i] * param1) + (f[i+1] * param2)$.

ELine Class

Represents a model of a line segment in EasyGauge.

Base Class: [EFrame](#)

PROPERTIES**End**

End point coordinates of the ELine object.

Length

Length of the ELine object.

Org**M** Origin point coordinates of the ELine object.**E****THODS****CopyTo**

Copies all the data of the current ELine object into another ELine object and returns it.

ELine

Constructs a ELine object.

<code>GetAngleBetweenLines</code>	Computes the angle between two lines.
<code>GetDistanceBetweenPointAndLine</code>	Computes the distance between a point and a line.
<code>GetIntersectionOfLines</code>	Computes the intersection between two lines.
<code>GetPoint</code>	Returns the coordinates of a point along the line.
<code>GetProjectionOfPointOnLine</code>	Computes the projection of a point on a line.
<code>operator=</code>	Copies all the data from another ELine object into the current ELine object
<code>Serialize</code>	-
<code>SetFromOriginAndEnd</code>	Sets the geometric parameters (center coordinates, length, and rotation angle) of a ELine object.
<code>SetFromTwoPoints</code>	E- L
<code>ine.CopyTo</code>	Copies all the data of the current ELine object into another ELine object and returns it.

[VB6]

```
ELine CopyTo(
    ELine other
)
```

Parameters

other

Pointer to the ELine object in which the current ELine object data have to be copied.

Remarks

In case of a **NULL** pointer, a new ELine object will be created and returned.

ELine.ELine

Constructs a ELine object.

[VB6]

```
void ELine(
)

void ELine(
    EPoint center,
    Single length,
    Single angle
)

void ELine(
    EPoint origin,
    EPoint end
)

void ELine(
    ELine other
)
```

Parameters

center

Center coordinates of the line at its nominal position. The default value is **(0,0)**.

length

Nominal length of the line. The default value is **100**.

angle

Nominal rotation angle of the line. The default value is **0**.

origin

Origin point coordinates of the line.

end

End point coordinates of the line.

other

Another ELine object to be copied in the new ELine object.

ELine.End

End point coordinates of the ELine object.

[VB6]

End As EPoint

read-only

ELine.GetAngleBetweenLines

Computes the angle between two lines.

[VB6]

```
Single GetAngleBetweenLines(
    ELine line1,
    ELine line2
)
```

Parameters

line1

First line

line2

Second line

Remarks

The angle returned by this function is signed in the trigonometric sense, meaning that angle (1,2) = -angle(2,1).

ELine.GetDistanceBetweenPointAndLine

Computes the distance between a point and a line.

```
[VB6]
Single GetDistanceBetweenPointAndLine(
    EPoint pt,
    ELine line,
    Boolean limited
)
```

Parameters

pt

The point.

line

The line.

limited

Indicates if the line parameter should be considered as an infinite line or as a segment.

ELine.GetIntersectionOfLines

Computes the intersection between two lines.

```
[VB6]
```

```
Long GetIntersectionOfLines(
    ELine line1,
    ELine line2,
    EPoint intersection,
    Boolean limited
)
```

Parameters*line1*

First line.

line2

Second line.

intersection

Found intersection.

limited

Indicates if the line parameters should be considered as infinite lines or as a segments.

Remarks

The function returns the number of intersections found. It will return -1 if the two lines are overlapping.

ELine.GetPoint

Returns the coordinates of a point along the line.

[VB6]

```
EPoint GetPoint(
    Single fraction
)
```

Parameters*fraction*

Point location expressed as a fraction of the line length (range [-1, +1]).

ELine.GetProjectionOfPointOnLine

Computes the projection of a point on a line.

[VB6]

```
EPoint GetProjectionOfPointOnLine(  
    EPoint pt,  
    ELine line  
)
```

Parameters

pt

The point.

line

The line.

ELine.Length

Length of the ELine object.

[VB6]

Length As Single

read-write

Remarks

By default, the length of the line is **100**, which means 100 pixels when the field of view is not calibrated, and 100 physical units in case of a calibrated field of view.

ELine.operator=

Copies all the data from another ELine object into the current ELine object

[VB6]

```
ELine operator=(  
    ELine other  
)
```

Parameters

other

ELine object to be copied

ELine.Org

Origin point coordinates of the ELine object.

[VB6]

```
Org As EPoint  
read-only
```

ELine.Serialize

-

[VB6]

```
void Serialize(
    ESerializer serializer,
    Long un32FileVersion
)
```

Parameters

serializer
-
un32FileVersion
-

ELine.SetFromOriginAndEnd

Sets the geometric parameters (center coordinates, length, and rotation angle) of a ELine object.

[VB6]

```
void SetFromOriginAndEnd(
    EPoint origin,
    EPoint end
)
```

Parameters

origin
Origin point coordinates of the line.
end
End point coordinates of the line.

ELine.SetFromTwoPoints

[VB6]

```
void SetFromTwoPoints(  
    EPoint origin,  
    EPoint end  
)
```

Parameters

origin

-

end

-

ELineGauge Class

Manages a line fitting gauge.

Base Class: [ELineShape](#)

PROPERTIES

Active

Sets the flag indicating whether the gauge is active or not.

AverageDistance

-

ClippingMode

Clipping mode, that allows to choose how the fitted segment length and center are computed.

FilteringThreshold

-

HVConstraint	-
KnownAngle	Flag indicating whether the slope of the line to be fitted is known or not.
Line	Sets the nominal position, length and rotation angle of the line fitting gauge, according to a known ELine object.
MeasuredLine	Information pertaining to the fitted line.
MinAmplitude	-
MinArea	-
NumFilteringPasses	-
NumMeasuredPoints	Number of edge-crossing points along the sample path of the gauge that was inspected with the last call to ELineGauge::MeasureSample .
NumSamples	-
NumSkipRanges	-
NumValidSamples	-

RectangularSamplingArea	-
SamplingStep	-
Shape	-
Smoothing	-
Thickness	-
Threshold	-
Tolerance	Searching area half thickness of the line fitting gauge.
TransitionChoice	-
TransitionIndex	-
TransitionType	-
Type	Shape type.
Valid	Flag indicating if at least one valid transition has been found.

METHODS

AddSkipRange

Adds an item to the set of skip ranges and returns the index of the newly added range.

CopyTo

Copies all the data of the current ELineGauge object into another ELineGauge object, and returns it.

Drag

Moves a handle to a new position and updates the position parameters of the gauge.

Draw

Draws a graphical representation of a point location or model fitting gauge, as defined by [EDrawingMode](#).

DrawWithCurrentPen

Draws a graphical representation of a point location or model fitting gauge, as defined by [EDrawingMode](#).

ELineGauge

Constructs a line measurement context.

GetMeasuredPeak

Returns information pertaining to the default derivative peak, along one of the sample paths of the gauge.

GetMeasuredPoint	Returns the coordinates of a sample point, measured along one of the sample paths of the gauge.
GetMinNumFitSamples	Returns the minimum number of samples required for fitting on each side of the shape.
GetSample	Allows to retrieve the sample points found along the line.
GetSkipRange	Allows to retrieve the start and end values of the skip range corresponding to the given index, if it is valid (i.e. if it has previously been created by a call to the ELineGauge::AddSkipRange method).
HitTest	Checks whether the cursor is positioned over a handle (TRUE) or not (FALSE).
Measure	Triggers the point location or the model fitting operation.
MeasureSample	Computes the sample points along the sample path whose index in the list is given by the pathIndex parameter.
MeasureWithoutFitting	Triggers the point location without line fitting operation.
operator=	Copies all the data from another ELineGauge object into the current ELineGauge object

Plot

Draws the profile that is crossed by one of the sample paths of the gauge, as defined by [EPlotItem](#).

PlotWithCurrentPen

Draws the profile that is crossed by one of the sample paths of the gauge, as defined by [EPlotItem](#).

Process

Triggers the process pertaining to a shape or gauge and all the daughter gauges attach to it.

RemoveAllSkipRanges

Removes all the skip ranges previously created by a call to [ELineGauge::AddSkipRange](#).

RemoveSkipRange

After a call to [ELineGauge::AddSkipRange](#), removes the skip range with the given index.

SetMinNumFitSamples

Sets the minimum number of samples required for fitting on each side of the shape.

ineGauge.Active

Sets the flag indicating whether the gauge is active or not.

[VB6]

Active As Boolean

read-write

Remarks

When complex gauging is required, several gauges can be grouped together. Applying [ELineGauge::Process](#) to the mother gauge or shape triggers the measurement of the whole. Only the active gauges will participate in the process. By default, the gauge is active (**TRUE**).

ELineGauge.AddSkipRange

Adds an item to the set of skip ranges and returns the index of the newly added range.

[VB6]

```
Long AddSkipRange(
    Long start,
    Long end
)
```

Parameters*start*

Beginning of the skip range.

end

End of the skip range.

Remarks

The samples indices between start and end (including the boundaries) will be discarded during the measurement process. The [AddSkipRange](#) method allows to define skip ranges in an [ELineGauge](#). This means that, at measure time, samples belonging to these ranges will not be taken into account. A sample may belong to more than one skip range; to be discarded, a sample has to pertain to at least one range. Moreover, the skip ranges are allowed to overlap one another. The range is allowed to be reversed (i.e. end is not required to be greater than start). Also, start and end are not required to reference valid indices at the time of the call (i.e. the range may lie outside of the current return value for [ELineGauge::NumSamples](#)).

ELineGauge.AverageDistance

-

[VB6]

AverageDistance As Single

read-only

ELineGauge.ClippingMode

Clipping mode, that allows to choose how the fitted segment length and center are computed.

[VB6]

ClippingMode As EClippingMode

read-write

Remarks

By default, the clipping mode is [EClippingMode_CenteredNominal](#), which corresponds to the behavior appearing in Open eVision version 6.4 and before.

ELineGauge.CopyTo

Copies all the data of the current ELineGauge object into another ELineGauge object, and returns it.

[VB6]

```
ELineGauge CopyTo(
    ELineGauge other,
    Boolean recursive
)
```

Parameters

other

Pointer to the ELineGauge object in which the current ELineGauge object data have to be copied.

recursive

TRUE if the children gauges have to be copied as well, **FALSE** otherwise.

Remarks

In case of a **NULL** pointer, a new ELineGauge object will be created and returned.

ELineGauge.Drag

Moves a handle to a new position and updates the position parameters of the gauge.

[VB6]

```
void Drag(
    Long x,
    Long y
)
```

Parameters

x

Cursor current coordinates.

y

Cursor current coordinates.

ELineGauge.Draw

Draws a graphical representation of a point location or model fitting gauge, as defined by [EDrawingMode](#).

[VB6]

```
void Draw(
    EDrawAdapter graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)

void Draw(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)

void Draw(
    Long graphicContext,
    ERGBColor color,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by [EDrawingMode](#).

daughters

TRUE if the daughters gauges are to be displayed also.

color

The color in which to draw the overlay.

ELineGauge.DrawWithCurrentPen

Draws a graphical representation of a point location or model fitting gauge, as defined by [EDrawingMode](#).

[VB6]

```
void DrawWithCurrentPen(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by [EDrawingMode](#).

daughters

TRUE if the daughters gauges are to be displayed also.

ELineGauge.ELineGauge

Constructs a line measurement context.

[VB6]

```
void ELineGauge(
)
void ELineGauge(
    ELineGauge other
)
```

Parameters*other*

Another ELineGauge object to be copied in the new ELineGauge object.

Remarks

With the default constructor, all the parameters are initialized to their respective default values. With the copy constructor, the constructed line measurement context is based on a pre-existing [ELineGauge](#) object. The gauge children are also copied. Hierarchy copying through a copy constructor is always recursive. To disable this recursion, use instead the [ELineGauge::CopyTo](#) method.

ELineGauge.FilteringThreshold

[VB6]

FilteringThreshold As Single

read-write

ELineGauge.GetMeasuredPeak

Returns information pertaining to the default derivative peak, along one of the sample paths of the gauge.

[VB6]

```
EPeak GetMeasuredPeak (
Long index
<>)
```

Parameters*index*

This argument must be left unchanged from its default value, i.e. **~0 (= 0xFFFFFFFF)**.

Remarks

[ELineGauge::GetMeasuredPeak](#) returns the information about the derivative peak that meets the following two requirements: 1. It lies on the sample path inspected with the last call to [ELineGauge::MeasureSample](#), and 1. It is associated with the edge-crossing point along the latter sample path that is selected by the transition choice parameter (cf. [ELineGauge::TransitionChoice](#)).

Note. For this method to succeed, it is necessary to previously call [ELineGauge::MeasureSample](#).

ELineGauge.GetMeasuredPoint

Returns the coordinates of a sample point, measured along one of the sample paths of the gauge.

[VB6]

```
EPoint GetMeasuredPoint(
    Long index
)
```

Parameters

index

This argument must be left unchanged from its default value, i.e. **~0 (= 0xFFFFFFFF)**.

Remarks

These coordinates pertain to the world space; they are expressed in the reference frame to which the current ELineGauge object belongs. The gauging process uses a list of sample points to find the shape position and size that best fit a given image. These sample points are measured along the sample paths defined by the gauge geometry.

[ELineGauge::GetMeasuredPoint](#) returns the coordinates of the sample point that meets the following two requirements: 1. It lies on the sample path inspected with the last call to [ELineGauge::MeasureSample](#), and 1. Among all the sample points along the latter sample path, it is the one selected by the [ELineGauge::TransitionChoice](#) property.

Note. For this method to succeed, it is necessary to previously call [ELineGauge::MeasureSample](#).

ELineGauge.GetMinNumFitSamples

Returns the minimum number of samples required for fitting on each side of the shape.

```
[VB6]  
void GetMinNumFitSamples(  
    Long side0,  
    Long side1,  
    Long side2,  
    Long side3  
)
```

Parameters

side0

Minimum number of samples on the top side of the rectangle.

side1

Minimum number of samples on the left side of the rectangle.

side2

Minimum number of samples on the bottom side of the rectangle.

side3

Minimum number of samples on the right side of the rectangle.

Remarks

Irrelevant in case of a point location operation.

ELineGauge.GetSample

Allows to retrieve the sample points found along the line.

```
[VB6]
```

```
Boolean GetSample(
    EPoint pt,
    Long index
)
```

Parameters*pt***EPoint** structure that will contain the sample position.*index*

The sample index

Remarks

The method provides the sample point corresponding to the supplied index. The returned value corresponds to the sample validity.

ELineGauge.GetSkipRange

Allows to retrieve the start and end values of the skip range corresponding to the given index, if it is valid (i.e. if it has previously been created by a call to the [ELineGauge::AddSkipRange](#) method).

[VB6]

```
void GetSkipRange(
    Long index,
    Long start,
    Long end
)
```

Parameters*index*

Index of the skip range.

start

Beginning of the skip range.

end

End of the skip range.

Remarks

Start is guaranteed to be smaller or equal to end.

ELineGauge.HitTest

Checks whether the cursor is positioned over a handle (**TRUE**) or not (**FALSE**).

[VB6]

```
Boolean HitTest(  
    Boolean daughters  
)
```

Parameters

daughters

TRUE if the daughters gauges handles have to be considered as well.

ELineGauge.HVConstraint

-

[VB6]

```
HVConstraint As Boolean  
read-write
```

ELineGauge.KnownAngle

Flag indicating whether the slope of the line to be fitted is known or not.

[VB6]

KnownAngle As Boolean

read-write

Remarks

A line model to be fitted may have a well-known slope. It is possible to impose the value of this slope, thus removing one degree of freedom. The line fitting gauge slope is set by means of [ELineShape::Angle](#). The sign of the rotation angle depends whether the field of view is calibrated or not. * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value. * When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

ELineGauge.Line

Sets the nominal position, length and rotation angle of the line fitting gauge, according to a known [ELine](#) object.

[VB6]

Line As ELine

read-write

ELineGauge.Measure

Triggers the point location or the model fitting operation.

[VB6]

```
void Measure(
    EROIBW8 sourceImage
)
```

Parameters

sourceImage

Pointer to the source image.

Remarks

When this method is called, and if not enough valid points are detected, then the method returns directly, and the measured gauge is set to the nominal parameters.

ELineGauge.MeasuredLine

Information pertaining to the fitted line.

[VB6]

MeasuredLine As ELine

read-only

ELineGauge.MeasureSample

Computes the sample points along the sample path whose index in the list is given by the **pathIndex** parameter.

[VB6]

```
void MeasureSample(
    EROIBW8 sourceImage,
    Long pathIndex
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

pathIndex

Sample path index.

Remarks

This method stores its results into a temporary variable inside the ELineGauge object.

ELineGauge.MeasureWithoutFitting

Triggers the point location without line fitting operation.

[VB6]

```
void MeasureWithoutFitting(
    EROIBW8 sourceImage
)
```

Parameters

sourceImage

Source image.

Remarks

This method performs the actual measurement for each transition, but does not perform the line fitting. This means that individual samples will be available through the [ELineGauge::GetSample](#) method, but the gauge position will not be changed. Please note that the filtering will not be performed in this method, since it relies upon the fitting process. The filtering parameters will thus be unused.

ELineGauge.MinAmplitude

[VB6]

MinAmplitude As Long

read-write

ELineGauge.MinArea

-

[VB6]

MinArea As Long

read-write

ELineGauge.NumFilteringPasses

-

[VB6]

NumFilteringPasses As Long

read-write

ELineGauge.NumMeasuredPoints

Number of edge-crossing points along the sample path of the gauge that was inspected with the last call to [ELineGauge::MeasureSample](#).

[VB6]

NumMeasuredPoints As Long

read-only

Remarks

Note. For this method to succeed, it is necessary to previously call [ELineGauge::MeasureSample](#).

ELineGauge.NumSamples

-

[VB6]

NumSamples As Long

read-only

ELineGauge.NumSkipRanges

-

[VB6]

NumSkipRanges As Long

read-only

ELineGauge.NumValidSamples

-

[VB6]

NumValidSamples As Long

read-only

ELineGauge.operator=

Copies all the data from another ELineGauge object into the current ELineGauge object

[VB6]

```
ELineGauge operator=(  
    ELineGauge other  
)
```

Parameters

other

ELineGauge object to be copied

ELineGauge.Plot

Draws the profile that is crossed by one of the sample paths of the gauge, as defined by [EPlotItem](#).

[VB6]

```
void Plot(
    Long graphicContext,
    EPlotItem drawItems,
    Single width,
    Single height,
    Single originX,
    Single originY
)

void Plot(
    Long graphicContext,
    ERGBColor color,
    EPlotItem drawItems,
    Single width,
    Single height,
    Single originX,
    Single originY
)

void Plot(
    EDrawAdapter graphicContext,
    EPlotItem drawItems,
    Single width,
    Single height,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawItems

Boolean combination of [EPlotItem](#) members, that indicates which items must be displayed.

width

Width of the plot.

height

Height of the plot.

originX

Origin point coordinates of the plot along the X axis.

originY

Origin point coordinates of the plot along the Y axis.

color

The color in which to draw the overlay.

Remarks

The sample path that is taken into considered is the one inspected with the last call to [ELineGauge::MeasureSample](#).

Note. For this method to succeed, it is necessary to previously call [ELineGauge::MeasureSample](#).

ELineGauge.PlotWithCurrentPen

Draws the profile that is crossed by one of the sample paths of the gauge, as defined by [EPlotItem](#).

[VB6]

```
void PlotWithCurrentPen(
    Long graphicContext,
    EPlotItem drawItems,
    Single width,
    Single height,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawItems

Boolean combination of [EPlotItem](#) members, that indicates which items must be displayed.

width

Width of the plot.

height

Height of the plot.

originX

Origin point coordinates of the plot along the X axis.

originY

Origin point coordinates of the plot along the Y axis.

Remarks

The sample path that is taken into considered is the one inspected with the last call to [ELineGauge::MeasureSample](#).

Note. For this method to succeed, it is necessary to previously call [ELineGauge::MeasureSample](#).

ELineGauge.Process

Triggers the process pertaining to a shape or gauge and all the daughter gauges attach to it.

[VB6]

```
void Process(
    EROIBW8 sourceImage,
    Boolean daughters
)
```

Parameters

sourceImage

Pointer to the source image.

daughters

Flag indicating whether the daughters shapes inherit of the same behavior.

Remarks

When complex gauging is required, several gauges can be grouped together. Applying **Process** to the mother gauge or shape triggers the measurement of the whole. Only the active gauges will participate in the process.

ELineGauge.RectangularSamplingArea

[VB6]

RectangularSamplingArea As Boolean

read-write

ELineGauge.RemoveAllSkipRanges

Removes all the skip ranges previously created by a call to [ELineGauge::AddSkipRange](#).

[VB6]

```
void RemoveAllSkipRanges()  
{}
```

ELineGauge.RemoveSkipRange

After a call to [ELineGauge::AddSkipRange](#), removes the skip range with the given index.

[VB6]

```
void RemoveSkipRange(  
    Long index  
)
```

Parameters

index

Index of the skip range to remove, as returned by [ELineGauge::AddSkipRange](#).

ELineGauge.SamplingStep

[VB6]

SamplingStep As Single

read-write

ELineGauge.SetMinNumFitSamples

Sets the minimum number of samples required for fitting on each side of the shape.

[VB6]

```
void SetMinNumFitSamples(  
    Long side0,  
    Long side1,  
    Long side2,  
    Long side3  
)
```

Parameters

side0

Required number of samples to correctly fit the line. The default value is **2**. It is the only parameter taken into account.

side1

Not used.

side2

Not used.

side3

Not used.

Remarks

Irrelevant in case of a point location operation. When the [ELineGauge::Measure](#) method is called, and if not enough valid points are detected, then the method returns directly, and the measured gauge is set to the nominal parameters.

ELineGauge.Shape

-

[VB6]

Shape As ELine

read-only

ELineGauge.Smoothing

-

[VB6]

Smoothing As Long

read-write

ELineGauge.Thickness

-

[VB6]

Thickness As Long

read-write

ELineGauge.Threshold

-

[VB6]

Threshold As Long

read-write

ELineGauge.Tolerance

Searching area half thickness of the line fitting gauge.

[VB6]

Tolerance As Single

read-write

Remarks

By default, the searching area thickness of the line fitting gauge is **20** (2x10), which means 20 pixels when the field of view is not calibrated, and 20 physical units in case of a calibrated field of view.

ELineGauge.TransitionChoice

-

[VB6]

TransitionChoice As ETransitionChoice

read-write

ELineGauge.TransitionIndex

-

[VB6]

TransitionIndex As Long

read-write

ELineGauge.TransitionType

-

[VB6]

TransitionType As ETransitionType

read-write

ELineGauge.Type

Shape type.

[VB6]

Type As EShapeType

read-only

ELineGauge.Valid

Flag indicating if at least one valid transition has been found.

[VB6]

Valid As Boolean

read-only

Remarks

A **FALSE** value means that no measurement has been performed. A **TRUE** value means that a transition was found along the sample path inspected with the last call to **ELineGauge::MeasureSample**, and thus a point has been measured.

ELineShape Class

-

Base Class: [EShape](#)**Derived Class(es):** [ELineGauge](#)

PROPERTIES

Angle

-

Center

-

CenterX	-
CenterY	-
End	-
Length	-
Line	-
Org	-
Scale	-
Type	M Shape type. E
THODS	
Closest	Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use EShape::ClosestShape .
CopyTo	-
Drag	-

Draw	Draws a graphical representation of a shape, as defined by EDrawingMode .
DrawWithCurrentPen	Draws a graphical representation of a shape, as defined by EDrawingMode .
GetPoint	Returns the coordinates of a point along the line.
HitTest	-
operator=	Copies all the data from another ELineShape object into the current ELineShape object
SetCenterXY	Sets the center coordinates of a ELineShape object.
SetFromOriginAndEnd	-
SetFromTwoPoints	E- L
ineShape.Angle	-

[VB6]

Angle As Single

read-write

ELineShape.Center

-

[VB6]

Center As EPoint

read-write

ELineShape.CenterX

-

[VB6]

CenterX As Single

read-only

ELineShape.CenterY

-

[VB6]

CenterY As Single

read-only

ELineShape.Closest

Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use [EShape::ClosestShape](#).

[VB6]

```
void Closest(  
)
```

ELineShape.CopyTo

[VB6]

```
ELineShape CopyTo(  
    ELineShape dest,  
    Boolean bRecursive  
)
```

Parameters

dest

bRecursive

ELineShape.Drag

-

[VB6]

```
void Drag(
    Long n32CursorX,
    Long n32CursorY
)
```

Parameters

n32CursorX

n32CursorY

ELineShape.Draw

Draws a graphical representation of a shape, as defined by [EDrawingMode](#).

[VB6]

```
void Draw(
    EDrawAdapter graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)

void Draw(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

```
void Draw(
    Long graphicContext,
    ERGBColor color,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by [EDrawingMode](#).

daughters

TRUE if the daughters gauges are to be displayed also.

color

-

ELineShape.DrawWithCurrentPen

Draws a graphical representation of a shape, as defined by [EDrawingMode](#).

[VB6]

```
void DrawWithCurrentPen(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by [EDrawingMode](#).

daughters

TRUE if the daughters gauges are to be displayed also.

ELineShape.End

-

[VB6]

End As EPoint

read-only

ELineShape.GetPoint

Returns the coordinates of a point along the line.

[VB6]

```
EPoint GetPoint(  
    Single fraction  
)
```

Parameters

fraction

Point location expressed as a fraction of the line length (range **[-1, +1]**).

ELineShape.HitTest

-

[VB6]

```
Boolean HitTest(  
    Boolean bDaughters  
)
```

Parameters

bDaughters

ELineShape.Length

-

Length As Single

read-write

ELineShape.Line

-

[VB6]

Line As ELine

read-write

ELineShape.operator=

Copies all the data from another ELineShape object into the current ELineShape object

[VB6]

```
ELineShape operator=(  
    ELineShape other  
)
```

Parameters

other

ELineShape object to be copied

ELineShape.Org

-

[VB6]

Org As EPoint

read-only

ELineShape.Scale

-

[VB6]

Scale As Single

read-write

ELineShape.SetCenterXY

Sets the center coordinates of a ELineShape object.

```
[VB6]  
  
void SetCenterXY(  
    Single centerX,  
    Single centerY  
)
```

Parameters

centerX
Center coordinates of the ELineShape object.
centerY
Center coordinates of the ELineShape object.

ELineShape.SetFromOriginAndEnd

```
[VB6]  
  
void SetFromOriginAndEnd(  
    EPoint origin,  
    EPoint end  
)
```

Parameters

origin
-
end

ELineShape.SetFromTwoPoints

—

[VB6]

```
void SetFromTwoPoints(  
    EPoint origin,  
    EPoint end  
)
```

Parameters

origin

-

end

-

ELineShape.Type

Shape type.

[VB6]

Type As EShapeType

read-only

EListIItem Class

Describes list items. This class pertains to the EasyObject legacy API. Please use [ECodedImage2](#) for all new developments instead.

Remarks

A list is a sequence of orderly list items. Each list item contains a pointer to a memory zone containing its data, a pointer to the previous list item, and a pointer to the next list item. List itemsA few [ECodedImage](#) methods handle EListIItem objects, or EListIItem pointers. Runs lists[ECodedImage::GetFirstRunData](#), [ECodedImage::GetFirstRunPtr](#),
[ECodedImage::GetLastRunData](#), [ECodedImage::GetLastRunPtr](#),
[ECodedImage::GetPreviousRunData](#), [ECodedImage::GetPreviousRunPtr](#),
[ECodedImage::GetNextRunData](#), [ECodedImage::GetNextRunPtr](#) These properties and methods allow to traverse the runs lists from the first run to the last, or from one run to its previous or next neighbor. A run can also be directly reached by its index within the list. The first run has index **0**. The last run has index **NumRuns-1**. The [ECodedImage::GetRunData](#) and [ECodedImage::GetRunDataPtr](#) methods return the run data, or a pointer to the run data. Objects lists[ECodedImage::GetFirstObjData](#), [ECodedImage::GetLastObjData](#),
[ECodedImage::GetPreviousObjData](#), [ECodedImage::GetPreviousObjPtr](#),
[ECodedImage::GetNextObjData](#), [ECodedImage::GetNextObjPtr](#) These properties and methods allow to traverse the objects lists from the first object to the last, or from one object to its previous or next neighbor. An object can also be directly reached by its index within the list. The first object has index **0**. The last object has index **NumObjects-1**. The [ECodedImage::GetObjectData](#) and [ECodedImage::GetObjDataPtr](#) methods return the object data, or a pointer to the object data.

EMatcher Class

Manages a complete matching context in EasyMatch.

Remarks

A matching context consists of a learned pattern and of the parameters required to locate one or more instances of the pattern in a search field.

PROPERTIES

AngleStep	Current angle step.
---------------------------	---------------------

ContrastMode	Contrast mode.
------------------------------	----------------

CorrelationMode	Correlation mode.
---------------------------------	-------------------

DontCareThreshold	"Don't care" threshold.
FilteringMode	Filtering mode.
FinalReduction	Index of the last reduction.
InitialMinScore	Minimum score applied as a selection criterion in the early stages of the matching process.
Interpolate	Interpolation mode.
IsotropicScale	Flag indicating whether isotropic (as opposed to anisotropic) scaling is used.
MaxAngle	Maximum angle, in the current angle unit.
MaxInitialPositions	Maximum number of positions at the first stage of the matching process.
MaxPositions	Maximum number of positions.
MaxScale	Maximum scale factor for isotropic scaling.
MaxScaleX	Maximum horizontal scale factor for anisotropic scaling.

MaxScaleY	Maximum vertical scale factor for anisotropic scaling.
MinAngle	Minimum angle, in the current angle unit.
MinReducedArea	Minimum reduced area parameter.
MinScale	Minimum scale factor for isotropic scaling.
MinScaleX	Minimum horizontal scale factor for anisotropic scaling.
MinScaleY	Minimum vertical scale factor for anisotropic scaling.
MinScore	Minimum score.
NumPositions	Number of good matches found, as defined by EMatcher::MinScore and EMatcher::MaxPositions properties.
NumReductions	Number of reduction steps used in the matching process.
PatternHeight	Learnt pattern height.

PatternLearnt	Returns TRUE after a learning operation has been successfully performed, indicating that the EMatcher object is ready for matching, and FALSE otherwise.
PatternType	-
PatternWidth	Learnt pattern width.
ScaleStep	Current value of scale step.
ScaleXStep	Current value of scale X step.
ScaleYStep	Current value of scale Y step.
Version	M Version number of the EMatcher object. E
THODS	
ClearImage	Releases the pointer to the image object that has been passed to the EMatcher object.
CopyLearntPattern	Copies the learnt pattern in the supplied image. If no pattern has been learned, an exception with code EError_NoPatternLearnt will be thrown.
CopyTo	Copies all the data of the current EMatcher object into another EMatcher object and returns it.

DrawPosition	Draws a graphical representation of a given occurrence of the pattern in the image, using a rectangle and possibly a small line segment in the upper-left corner.
DrawPositions	Draws a graphical representation of all occurrences of the pattern in the image, using a rectangle and possibly a small line segment in the upper-left corner.
DrawPositionsWithCurrentPen	Draws a graphical representation of all occurrences of the pattern in the image, using a rectangle and possibly a small line segment in the upper-left corner.
DrawPositionWithCurrentPen	Draws a graphical representation of a given occurrence of the pattern in the image, using a rectangle and possibly a small line segment in the upper-left corner.
EMatcher	Constructs a matching context.
GetPixelDimensions	Gets the physical pixel dimensions.
GetPosition	Returns an EMatchPosition object containing the position coordinates.
LearnPattern	Learns a pattern to subsequently match in an image.
Load	-

Match	Matches the pattern against an image.
operator=	Copies all the data from another EMatcher object into the current EMatcher object
Save	-
SetExtension	Sets the extension of the matching ROI, i.e. puts the horizontal and vertical distances, in pixels, that the found pattern occurrences may fall outside the matching ROI. Such occurrences partially outside the ROI have their score corrected by the ratio between the pattern area outside the ROI and the pattern area inside.
SetPixelDimensions	E Sets the physical pixel dimensions.
	M
atcher.AngleStep	

Current angle step.

[VB6]

AngleStep As Single

read-only

EMatcher.ClearImage

Releases the pointer to the image object that has been passed to the EMatcher object.

[VB6]

```
void ClearImage()  
 )
```

Remarks

It is the way to tell to the EMatcher object that its pointer is not valid anymore. The [EMatcher::Match](#) method keeps a copy of the image pointer given as parameter. So, if the user deletes this pointer, the EMatcher object should be informed.

EMatcher.ContrastMode

Contrast mode.

[VB6]

```
ContrastMode As EMatchContrastMode  
read-write
```

Remarks

By default, the contrast mode is set to [EMatchContrastMode_Normal](#).

EMatcher.CopyLearntPattern

Copies the learnt pattern in the supplied image. If no pattern has been learned, an exception with code [EError_NoPatternLearnt](#) will be thrown.

```
[VB6]

void CopyLearnedPattern(
    EImageBW8 image
)

void CopyLearnedPattern(
    EImageC24 image
)
```

Parameters

image

Pointer to the image in which the learnt pattern will be returned.

EMatcher.CopyTo

Copies all the data of the current EMatcher object into another EMatcher object and returns it.

```
[VB6]

EMatcher CopyTo(
    EMatcher other
)
```

Parameters

other

Pointer to the EMatcher object in which the current EMatcher object parameters are to be copied. If **NULL** (default), a new EMatcher object will be created and returned.

EMatcher.CorrelationMode

Correlation mode.

[VB6]

CorrelationMode As ECorrelationMode

read-write

Remarks

This property tells what normalization rule is used to correlate the pattern to the image. By default, the correlation mode is set to [ECorrelationMode_Normalized](#).

EMatcher.DontCareThreshold

"Don't care" threshold.

[VB6]

DontCareThreshold As Long

read-write

Remarks

If the pattern cannot be inscribed in a rectangle because there are foreign objects in a close neighborhood, mismatches can be avoided by using "don't care" pixels: all the pattern pixels whose value is strictly below **DontCareThreshold** will be ignored. By default, this property is set to **0**: no "don't care" pixel exists.

Note. When you use the "don't care" feature, either the pattern is well contrasted from its background -then set **DontCareThreshold** to an appropriate thresholding value- or it is not - then set the background pixels of the pattern to some low value (by a masking operation) and set **DontCareThreshold** to this low value plus one.

EMatcher.DrawPosition

Draws a graphical representation of a given occurrence of the pattern in the image, using a rectangle and possibly a small line segment in the upper-left corner.

[VB6]

```

void DrawPosition(
    Long graphicContext,
    Long index,
    Boolean bCorner,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawPosition(
    Long graphicContext,
    ERGBColor color,
    Long index,
    Boolean bCorner,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawPosition(
    EDrawAdapter graphicContext,
    Long index,
    Boolean bCorner,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

```

Parameters

graphicContext

Handle of the device context on which to draw.

index

Occurrence index, in range **0..NumPositions-1**.

bCorner

TRUE if the corner mark is to be drawn. **FALSE** by default. (This mark is useful when large rotations are allowed.)

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used.

EMatcher.DrawPositions

Draws a graphical representation of all occurrences of the pattern in the image, using a rectangle and possibly a small line segment in the upper-left corner.

[VB6]

```
void DrawPositions(
    Long graphicContext,
    Boolean bCorner,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawPositions(
    Long graphicContext,
    ERGBColor color,
    Boolean bCorner,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

```
void DrawPositions(
    EDrawAdapter graphicContext,
    Boolean bCorner,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

bCorner

TRUE if the corner mark is to be drawn. **FALSE** by default. (This mark is useful when large rotations are allowed.)

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used for all occurrences (to vary the colors, draw the objects separately using the [EMatcher::DrawPosition](#) method instead).

EMatcher.DrawPositionsWithCurrentPen

Draws a graphical representation of all occurrences of the pattern in the image, using a rectangle and possibly a small line segment in the upper-left corner.

[VB6]

```
void DrawPositionsWithCurrentPen(
    Long graphicContext,
    Boolean bCorner,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

bCorner

TRUE if the corner mark is to be drawn. **FALSE** by default. (This mark is useful when large rotations are allowed.)

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used for all occurrences (to vary the colors, draw the objects separately using the [EMatcher::DrawPosition](#) method instead).

EMatcher.DrawPositionWithCurrentPen

Draws a graphical representation of a given occurrence of the pattern in the image, using a rectangle and possibly a small line segment in the upper-left corner.

[VB6]

```
void DrawPositionWithCurrentPen(
    Long graphicContext,
    Long index,
    Boolean bCorner,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

index

Occurrence index, in range **0..NumPositions-1**.

bCorner

TRUE if the corner mark is to be drawn. **FALSE** by default. (This mark is useful when large rotations are allowed.)

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used.

EMatcher.EMatcher

Constructs a matching context.

```
[VB6]

void EMatcher(
    )
void EMatcher(
    Long maximumNumberOfDegreesOfFreedom
    )
void EMatcher(
    EMatcher other
    )
```

Parameters

maximumNumberOfDegreesOfFreedom

-

other

-

Remarks

With the default constructor (no argument), all parameters are initialized to their respective default values. The copy constructor constructs a matching context based on a pre-existing EMatcher object. The last constructor constructs a matching context with a specified number of degrees of freedom. **maximumNumberOfDegreesOfFreedom** is the maximum number of degrees of freedom that the EMatcher object being constructed will be allowed to use during its life. All other parameters are initialized to their respective default values. The degrees of freedom for a matching operation are the *translation* (2 modes), the *rotation* (1 mode), the *isotropic scaling* (1 mode) and the *anisotropic scaling* (1 more mode). There is no way to modify this parameter for an existing EMatcher object. The default value for **maximumNumberOfDegreesOfFreedom** is **5**, that is the maximum value. The minimum value for the number of degrees of freedom is **2**, in order to allow, at least, the pattern translation.

EMatcher.FilteringMode

Filtering mode.

```
[VB6]
```

FilteringMode As EFilteringMode

read-write

Remarks

To achieve acceptable time performance, EasyMatch works by sub-sampling the pattern in the early phases of the processing. The filtering mode parameter allows to select the pre-processing type applied to the image before the decimation: averaging or low-pass filtering. By default, this property is set to [EFilteringMode_Uniform](#), whereas the [EFilteringMode_LowPass](#) mode is indicated if the image presents sharp gray-level transitions.

EMatcher.FinalReduction

Index of the last reduction.

[VB6]

FinalReduction As Long

read-write

Remarks

The pattern matching process is comprised of a few passes (typically 4) during which the position accuracy is improved by a factor of 2 (for all degrees of freedom). This is called a *reduction*. By default, the computation continues until an accuracy of one pixel is obtained. To speed up the process, the last reduction passes can be dropped, so lowering the accuracy. Anyway, the interpolation mode can then still be used. By default, this property is set to **0** (pixel accuracy). Value **1** corresponds to 2-pixels accuracy, **2** to 4, and so on. The range of values that can be used for this property is **0** to **NumReductions-1**.

EMatcher.GetPixelDimensions

Gets the physical pixel dimensions.

```
[VB6]  
void GetPixelDimensions(  
    Single width,  
    Single height  
)
```

Parameters

width

Width of a pixel.

height

Height of a pixel.

EMatcher.GetPosition

Returns an [EMatchPosition](#) object containing the position coordinates.

```
[VB6]  
EMatchPosition GetPosition(  
    Long index  
)
```

Parameters

index

0-based index to the desired position. The positions are ordered by decreasing score.

EMatcher.InitialMinScore

Minimum score applied as a selection criterion in the early stages of the matching process.

```
[VB6]
```

InitialMinScore As Single

read-write

Remarks

When the search for matches starts, EasyMatch considers a set of candidate positions that it progressively refines. When they later appear to be bad candidates, they are rejected. Though it is the minimum score level that is used to reject bad matching positions at the final step of the matching process, the "initial minimum score" parameter is used to eliminate bad positions (whose score is not high enough) in the early stages of the matching processing. If the matching process is achieved in one step, only the minimum score parameter will be considered. By default, this property is set to **-1**.

EMatcher.Interpolate

Interpolation mode.

[VB6]

Interpolate As Boolean

read-write

Remarks

By default, matching is done with a one-pixel precision for all degrees of freedom (translation, rotation and scaling). You can use an additional interpolation process to achieve sub-pixel accuracy. This generally leads to an improvement of the sub-pixel accuracy by a factor larger than 10. This is possible only when the found instances match closely the model. A score higher than 0.99 indicates that the instances are a close match of the model. In other words, the instance is considered to be more accurate when the score is higher. The added computational cost is low. By default, this property is set to **FALSE**.

EMatcher.IsotropicScale

Flag indicating whether isotropic (as opposed to anisotropic) scaling is used.

[VB6]

IsotropicScale As Boolean

read-only

Remarks

TRUE if isotropic (as opposed to anisotropic) scaling is used, i.e. when the scale factors in both the X and Y directions are equal.

EMatcher.LearnPattern

Learns a pattern to subsequently match in an image.

[VB6]

```
void LearnPattern(  
    EROIBW8 pattern  
)  
  
void LearnPattern(  
    EROIC24 pattern  
)
```

Parameters

pattern

Remarks

The maximum size for a pattern is 1791x1791.

EMatcher.Load

```
[VB6]
void Load(
    ESerializer serializer
)
void Load(
    String stream
)
```

Parameters

serializer

-

stream

-

EMatcher.Match

Matches the pattern against an image.

```
[VB6]
void Match(
    EROIBW8 image
)
void Match(
    EROIC24 image
)
```

Parameters

image

Pointer to the image/ROI within which the pattern will be searched for.

Remarks

The matching results can be obtained by means of the [EMatcher::NumPositions](#) and [EMatcher::GetPosition](#) members.

EMatcher.MaxAngle

Maximum angle, in the current angle unit.

[VB6]

MaxAngle As Single

read-write

Remarks

The rotation of the pattern is allowed within the range (**-1 <= MinAngle < MaxAngle <= 1** revolution). By default, both remain **0**.

EMatcher.MaxInitialPositions

Maximum number of positions at the first stage of the matching process.

[VB6]

MaxInitialPositions As Long

read-write

Remarks

When the search for matches starts, EasyMatch considers a set of candidate positions that it progressively refines. When they later appear to be bad candidates, they are rejected. Eventually, a maximum of [EMatcher::MaxPositions](#) is returned. In some circumstances, when the image contains features roughly similar to the pattern, these can confuse the matching process, resulting in false matches. To overcome this situation, increasing the number of initial positions will help. By default, this property is set to **0**, indicating that the value of [EMatcher::MaxPositions](#) should be used instead.

EMatcher.MaxPositions

Maximum number of positions.

[VB6]

MaxPositions As Long

read-write

Remarks

Indicates how many matching positions have to be returned at a maximum. This number corresponds to the expected maximum number of occurrences of the pattern (it is sometimes advisable to use a few additional positions). By default, this property is set to **1**.

EMatcher.MaxScale

Maximum scale factor for isotropic scaling.

[VB6]

MaxScale As Single

read-write

Remarks

Two scaling modes are allowed: in isotropic mode, the scale factor is identical in all directions; in anisotropic mode, the scale factors differ in the horizontal and vertical directions. To select the appropriate mode, it suffices to call the relevant **Set** member. The scaling of the pattern is allowed within the range (**0.5 <= MinScale < MaxScale <= 2**). By default, both remain **1**. The same holds for anisotropic scale factors.

EMatcher.MaxScaleX

Maximum horizontal scale factor for anisotropic scaling.

[VB6]

MaxScaleX As Single

read-write

Remarks

Two scaling modes are allowed: in isotropic mode, the scale factor is identical in all directions; in anisotropic mode, the scale factors differ in the horizontal and vertical directions. To select the appropriate mode, it suffices to call the relevant **Set** member. The scaling of the pattern is allowed within the range (**0.5 <= MinScaleX < MaxScaleX <= 2**). By default, both remain **1**. The same holds for anisotropic scale factors.

EMatcher.MaxScaleY

Maximum vertical scale factor for anisotropic scaling.

[VB6]

MaxScaleY As Single

read-write

Remarks

Two scaling modes are allowed: in isotropic mode, the scale factor is identical in all directions; in anisotropic mode, the scale factors differ in the horizontal and vertical directions. To select the appropriate mode, it suffices to call the relevant **Set** member. The scaling of the pattern is allowed within the range (**0.5 <= MinScaleY < MaxScaleY <= 2**). By default, both remain **1**. The same holds for anisotropic scale factors.

EMatcher.MinAngle

Minimum angle, in the current angle unit.

[VB6]

MinAngle As Single

read-write

Remarks

The rotation of the pattern is allowed within the range (**-1 <= MinAngle < MaxAngle <= 1** revolution). By default, both remain **0**.

EMatcher.MinReducedArea

Minimum reduced area parameter.

[VB6]

MinReducedArea As Long

read-write

Remarks

To achieve acceptable time performance, EasyMatch works by under-sampling the pattern in the early phases of the processing. This property tells how many pixels of the pattern are kept, at a minimum. By default, this property is set to **64**, which is the right choice in most situations. Higher values are not recommended. Lower values can speed up the processing, but sometimes cause the matching process fail to find the best matches. To circumvent this problem, you can use guard positions, that is find more matches than expected and keep the good ones only.

Note. Changing this property invalidates any previous learning.

EMatcher.MinScale

Minimum scale factor for isotropic scaling.

[VB6]

MinScale As Single

read-write

Remarks

Two scaling modes are allowed: in isotropic mode, the scale factor is identical in all directions; in anisotropic mode, the scale factors differ in the horizontal and vertical directions. To select the appropriate mode, it suffices to call the relevant **Set** member. The scaling of the pattern is allowed within the range (**0.5 <= MinScale < MaxScale <= 2**). By default, both remain **1**. The same holds for anisotropic scale factors.

EMatcher.MinScaleX

Minimum horizontal scale factor for anisotropic scaling.

[VB6]

MinScaleX As Single

read-write

Remarks

Two scaling modes are allowed: in isotropic mode, the scale factor is identical in all directions; in anisotropic mode, the scale factors differ in the horizontal and vertical directions. To select the appropriate mode, it suffices to call the relevant **Set** member. The scaling of the pattern is allowed within the range (**0.5 <= MinScaleX < MaxScaleX <= 2**). By default, both remain **1**. The same holds for anisotropic scale factors.

EMatcher.MinScaleY

Minimum vertical scale factor for anisotropic scaling.

[VB6]

MinScaleY As Single

read-write

Remarks

Two scaling modes are allowed: in isotropic mode, the scale factor is identical in all directions; in anisotropic mode, the scale factors differ in the horizontal and vertical directions. To select the appropriate mode, it suffices to call the relevant **Set** member. The scaling of the pattern is allowed within the range (**0.5 <= MinScaleY < MaxScaleY <= 2**). By default, both remain **1**. The same holds for anisotropic scale factors.

EMatcher.MinScore

Minimum score.

[VB6]

MinScore As Single

read-write

Remarks

This property indicates what score a match must reach to be considered as good, and to be returned in the list of positions; this selection criterion is applied at the final stage of the matching process. One good way to select the appropriate **EMatcher::MinScore** in a given context is to set it to **-1** (any match will be retained), set **EMatcher::MaxPositions** to more than needed, and examine the returned scores after a matching. By default, this property is set to **-1**.

EMatcher.NumPositions

Number of good matches found, as defined by [EMatcher::MinScore](#) and [EMatcher::MaxPositions](#) properties.

[VB6]

NumPositions As Long

read-only

EMatcher.NumReductions

Number of reduction steps used in the matching process.

[VB6]

NumReductions As Long

read-only

Remarks

These depend on the actual pattern size, and on the **MinReducedArea** property. The **FinalReduction** property, used to speed up matching when coarse location is sufficient, must be set in range **0..NumReductions-1**.

EMatcher.operator=

Copies all the data from another EMatcher object into the current EMatcher object

[VB6]

```
EMatcher operator=(  
    EMatcher other  
)
```

Parameters

other

EMatcher object to be copied

EMatcher.PatternHeight

Learnt pattern height.

[VB6]

PatternHeight As Long

read-only

EMatcher.PatternLearnt

Returns **TRUE** after a learning operation has been successfully performed, indicating that the EMatcher object is ready for matching, and **FALSE** otherwise.

[VB6]

PatternLearnt As Boolean

read-only

EMatcher.PatternType

[VB6]

PatternType As EImageType

read-only

EMatcher.PatternWidth

Learnt pattern width.

[VB6]

PatternWidth As Long

read-only

EMatcher.Save

-

-

-

void Save(
 ESerializer serializer
)

void Save(
 String stream
)

Parameters

serializer

-

stream

-

EMatcher.ScaleStep

Current value of scale step.

[VB6]

ScaleStep As Single

read-only

EMatcher.ScaleXStep

Current value of scale X step.

[VB6]

ScaleXStep As Single

read-only

EMatcher.ScaleYStep

Current value of scale Y step.

[VB6]

ScaleYStep As Single

read-only

EMatcher.SetExtension

Sets the extension of the matching ROI, i.e. puts the horizontal and vertical distances, in pixels, that the found pattern occurrences may fall outside the matching ROI. Such occurrences partially outside the ROI have their score corrected by the ratio between the pattern area outside the ROI and the pattern area inside.

```
[VB6]  
void SetExtension(  
    Long n32ExtensionX,  
    Long n32ExtensionY  
)
```

Parameters

n32ExtensionX

extension outside the matching ROI along its Width, in pixels.

n32ExtensionY

extension outside the matching ROI along its Height, in pixels.

EMatcher.SetPixelDimensions

Sets the physical pixel dimensions.

```
[VB6]  
void SetPixelDimensions(  
    Single width,  
    Single height  
)
```

Parameters

width

Width of a pixel.

height

Height of a pixel.

Remarks

When an image has been acquired in such a way that the pixels are "non-square" — the physical width and height of the area covered by a pixel are unequal — a form of anisotropy results. When rotated, the objects become skewed; rectangles become parallelograms. In such a situation, the pixel aspect ratio must be known to compensate it during matching. The aspect ratio is given by specifying the true width and height of a pixel. The specification of the pixel dimensions is only useful when rotation is used. Only the aspect ratio matters, so that relative width and height values can be given. By default, the pixel width and height are set to **1.0**.

EMatcher.Version

Version number of the EMatcher object.

[VB6]

Version As Long

read-only

EMatrixCode Class

Holds all the information regarding a single Data Matrix code: its decoded string, its grading, its errors,....

PROPERTIES

Angle

MatrixCode angle.

AxialNonUniformity

Measured axial non-uniformity value (**1.0** to **0** scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.

AxialNonUniformityGrade

Measured axial non-uniformity grading (**4** to **0** scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.

CellDefects

Measured cell defects value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.

Center

EMatrixCode center.

Contrast

Measured symbol contrast value, as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.

ContrastGrade

Measured symbol contrast grading (**4** to **0** scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.

ContrastType

Symbol contrast type, as defined by **EMatrixCodeContrastMode**.

DataMatrixCellHeight	Measured data matrix cell height value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.
DataMatrixCellWidth	Measured data matrix cell width value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.
DecodedString	Decoded Data Matrix symbol string.
Family	ECC symbol family, as defined by EFamily .
FinderPatternDefects	Measured finder pattern defects value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.
Flipping	Symbol flipping type, as defined by EFlipping .
Found	TRUE if a EMatrixCode object has been found in the ROI supplied to EMatrixCodeReader::Read , even if it could not be successfully decoded.
HorizontalMarkGrowth	Measured horizontal mark growth value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.

HorizontalMarkMisplacement	Measured horizontal mark misplacement value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.
Iso15415GradingParameters	ISO/IEC 15415 grading parameters
Iso29158GradingParameters	ISO/IEC 29158 grading parameters
LocationThreshold	Absolute threshold (0 to 255 scale) that was used during location of the symbol.
LogicalSize	Symbol logical size, as defined by ELogicalSize .
LogicalSizeHeight	For a logical size of S1xS2, LogicalSizeHeight is the integer S1.
LogicalSizeWidth	For a logical size of S1xS2, LogicalSizeWidth is the integer S2.
MeasuredPrintGrowth	Raw, un-normalized print quality parameter (1.0 to 0 scale), after a read operation, provided that the print quality assessment has been enabled.
NumErrors	Number of errors that were detected and corrected while decoding the symbol.

OverallGrade	Overall symbol grading (4 to 0 scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.
PrintGrowth	Normalized measured print growth value, as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.
PrintGrowthGrade	Measured print growth grading (4 to 0 scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.
ReadingThreshold	Absolute threshold (0 to 255 scale) that was used during reading of the symbol.
SemiT10GradingParameters	Semi T10-0701 grading parameters
SymbolContrastSNR	Measured symbol contrast signal to noise ratio value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.
UnusedErrorCorrection	Measured unused error correction value (1.0 to 0 scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.

UnusedErrorCorrectionGrade	Measured unused error correction grading (4 to 0 scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.
VerticalMarkGrowth	Measured vertical mark growth value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.
VerticalMarkMisplacement	M Measured vertical mark misplacement value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.
THODS	
Draw	Draws the EMatrixCode corner points and symbol finder pattern (when the symbol size has been correctly detected).
DrawErrors	Draws all symbol finder pattern cells where errors were detected and corrected.
DrawErrorsWithCurrentPen	Draws all symbol finder pattern cells where errors were detected and corrected.
DrawWithCurrentPen	Draws the EMatrixCode corner points and symbol finder pattern (when the symbol size has been correctly detected).
EMatrixCode	Constructs a EMatrixCode context.

GetCorner

Gets a matrix code corner.

GetDecodedDataElement

Gets a character value of the matrix code.

IsGS1**TRUE** if the decoded string uses the GS1 standard**Load**

-

operator=

Copies all the data from another EMatrixCode object into the current EMatrixCode object

Save

-

SetCorner

Sets a matrix code corner.

M

atrixCode.Angle

MatrixCode angle.

[VB6]

Angle As Single

read-only

Remarks

EMatrixCode.AxialNonUniformity

Measured axial non-uniformity value (**1.0** to **0** scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.

[VB6]

AxialNonUniformity As Single

read-only

EMatrixCode.AxialNonUniformityGrade

Measured axial non-uniformity grading (**4** to **0** scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.

[VB6]

AxialNonUniformityGrade As Long

read-only

EMatrixCode.CellDefects

Measured cell defects value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.

[VB6]

CellDefects As Single

read-only

Remarks

Read-only.

EMatrixCode.Center

[EMatrixCode](#) center.

[VB6]

Center As EPoint

read-only

EMatrixCode.Congtrast

Measured symbol contrast value, as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.

[VB6]

Contrast As Single

read-only

Remarks

The [EMatrixCode::Contrast](#) property is computed as the difference of the reference gray levels over their arithmetic average. Values range between **0** and **1.0**.

EMatrixCode.CongtrastGrade

Measured symbol contrast grading (**4** to **0** scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.

[VB6]

ContrastGrade As Long

read-only

EMatrixCode.ContrastType

Symbol contrast type, as defined by [EMatrixCodeContrastMode](#).

[VB6]

ContrastType As EMatrixCodeContrastMode

read-only

EMatrixCode.DataMatrixCellHeight

Measured data matrix cell height value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.

[VB6]

DataMatrixCellHeight As Single

read-only

Remarks

Read-only.

EMatrixCode.DataMatrixCellWidth

Measured data matrix cell width value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.

[VB6]

DataMatrixCellWidth As Single

read-only

Remarks

Read-only.

EMatrixCode.DecodedString

Decoded Data Matrix symbol string.

[VB6]

DecodedString As String

read-only

EMatrixCode.Draw

Draws the **EMatrixCode** corner points and symbol finder pattern (when the symbol size has been correctly detected).

[VB6]

```

void Draw(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void Draw(
    Long graphicContext,
    ERGBColor color,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void Draw(
    EDrawAdapter graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

```

Parameters

graphicContext

Handle of the device context on which to draw.

zoomX

Horizontal zooming factor. A value greater than **1** means zoom in. By default, **TRUE** scale is used.

zoomY

Vertical zooming factor. A value greater than **1** means zoom in. If set to **0**, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor, in pixels. By default, no panning occurs.

panY

Vertical panning factor, in pixels. By default, no panning occurs.

color

The color in which to draw the overlay.

Remarks

The reference corner has a bold cross marking.

EMatrixCode.DrawErrors

Draws all symbol finder pattern cells where errors were detected and corrected.

[VB6]

```
void DrawErrors(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawErrors(
    Long graphicContext,
    ERGBColor color,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawErrors(
    EDrawAdapter graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

zoomX

Horizontal zooming factor. A value greater than **1** means zoom in. By default, **TRUE** scale is used.

zoomY

Vertical zooming factor. A value greater than **1** means zoom in. If set to **0**, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor, in pixels. By default, no panning occurs.

panY

Vertical panning factor, in pixels. By default, no panning occurs.

color

The color in which to draw the overlay.

Remarks

This member is intended to be called in conjunction with [EMatrixCode::Draw](#).

EMatrixCode.DrawErrorsWithCurrentPen

Draws all symbol finder pattern cells where errors were detected and corrected.

[VB6]

```
void DrawErrorsWithCurrentPen(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

zoomX

Horizontal zooming factor. A value greater than **1** means zoom in. By default, **TRUE** scale is used.

zoomY

Vertical zooming factor. A value greater than **1** means zoom in. If set to **0**, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor, in pixels. By default, no panning occurs.

panY

Vertical panning factor, in pixels. By default, no panning occurs.

Remarks

This member is intended to be called in conjunction with [EMatrixCode::Draw](#).

EMatrixCode.DrawWithCurrentPen

Draws the **EMatrixCode** corner points and symbol finder pattern (when the symbol size has been correctly detected).

[VB6]

```
void DrawWithCurrentPen(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

zoomX

Horizontal zooming factor. A value greater than **1** means zoom in. By default, **TRUE** scale is used.

zoomY

Vertical zooming factor. A value greater than **1** means zoom in. If set to **0**, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor, in pixels. By default, no panning occurs.

panY

Vertical panning factor, in pixels. By default, no panning occurs.

Remarks

The reference corner has a bold cross marking.

EMatrixCode.EMatrixCode

Constructs a **EMatrixCode** context.

```
[VB6]
void EMATRIXCODE()
void EMATRIXCODE(EMATRIXCODE other)
```

Parameters

other

Another EMATRIXCODE object to be copied in the new EMATRIXCODE object.

Remarks

The default constructor constructs an uninitialized **EMATRIXCODE** object. All properties are initialized to their respective default values. The copy constructor constructs a **EMATRIXCODE** context based on a pre-existing **EMATRIXCODE** object. All properties and internal data are copied.

EMATRIXCODE.FAMILY

ECC symbol family, as defined by **EFamily**.

```
[VB6]
Family As EFamily
read-only
```

EMATRIXCODE.FINDERPATTERNDEFECTS

Measured finder pattern defects value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.

[VB6]

FinderPatternDefects As Single

read-only

Remarks

Read-only.

EMatrixCode.Flipping

Symbol flipping type, as defined by [EFlipping](#).

[VB6]

Flipping As EFlipping

read-only

EMatrixCode.Found

TRUE if a [EMatrixCode](#) object has been found in the ROI supplied to [EMatrixCodeReader::Read](#), even if it could not be successfully decoded.

[VB6]

Found As Boolean

read-only

Remarks

If this property is **FALSE**, it is still possible that an unlocalized matrix code exists in the image. However, if **Found** is **FALSE**, no other property should be read.

EMatrixCode.GetCorner

Gets a matrix code corner.

[VB6]

```
EPoint GetCorner(  
    Long index  
)
```

Parameters

index

Index of the matrix code corner.

EMatrixCode.GetDecodedDataElement

Gets a character value of the matrix code.

[VB6]

```
Byte GetDecodedDataElement(  
    Long index  
)
```

Parameters

index

Index of the character value.

Remarks

This property makes it possible to see information not coded as ASCII characters.

EMatrixCode.HorizontalMarkGrowth

Measured horizontal mark growth value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.

[VB6]

HorizontalMarkGrowth As Single

read-only

Remarks

Read-only.

EMatrixCode.HorizontalMarkMisplacement

Measured horizontal mark misplacement value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.

[VB6]

HorizontalMarkMisplacement As Single

read-only

Remarks

Read-only.

EMatrixCode.IsGS1

TRUE if the decoded string uses the GS1 standard

[VB6]

```
Boolean IsGS1(  
)
```

EMatrixCode.Iso15415GradingParameters

ISO/IEC 15415 grading parameters

[VB6]

```
Iso15415GradingParameters As EMatrixCodeIso15415GradingParameters  
read-only
```

Remarks

Read-only.

EMatrixCode.Iso29158GradingParameters

ISO/IEC 29158 grading parameters

[VB6]

```
Iso29158GradingParameters As EMatrixCodeIso29158GradingParameters  
read-only
```

Remarks

Read-only.

EMatrixCode.Load

-

[VB6]

```
void Load(  
    ESerializer serializer  
)  
  
void Load(  
    String stream  
)
```

Parameters

serializer

-

stream

-

EMatrixCode.LocationThreshold

Absolute threshold (0 to 255 scale) that was used during location of the symbol.

[VB6]

```
LocationThreshold As Long  
read-only
```

EMatrixCode.LogicalSize

Symbol logical size, as defined by [ELogicalSize](#).

[VB6]

LogicalSize As ELogicalSize

read-only

EMatrixCode.LogicalSizeHeight

For a logical size of S1xS2, **LogicalSizeHeight** is the integer S1.

[VB6]

LogicalSizeHeight As Long

read-only

EMatrixCode.LogicalSizeWidth

For a logical size of S1xS2, **LogicalSizeWidth** is the integer S2.

[VB6]

LogicalSizeWidth As Long

read-only

EMatrixCode.MeasuredPrintGrowth

Raw, un-normalized print quality parameter (**1.0** to **0** scale), after a read operation, provided that the print quality assessment has been enabled.

[VB6]

MeasuredPrintGrowth As Single

read-only

Remarks

The **EMatrixCode::MeasuredPrintGrowth** property is computed as the measured area of the active cells (same color as the finder pattern) over the area of ideal square cells, having sides equal to the pitches (such cells perfectly tile the symbol). Its value typically is on the order of **1.0**, regardless the symbol size.

EMatrixCode.NumErrors

Number of errors that were detected and corrected while decoding the symbol.

[VB6]

NumErrors As Long

read-only

Remarks

Such errors may be due to symbol degradation by scratches, blur, non-uniform illumination or slight changes in size.

EMatrixCode.operator=

Copies all the data from another EMatrixCode object into the current EMatrixCode object

[VB6]

```
EMatrixCode operator=(
EMatrixCode other
<>)
```

Parameters

other

EMatrixCode object to be copied

EMatrixCode.OverallGrade

Overall symbol grading (**4** to **0** scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.

[VB6]

OverallGrade As Long

read-only

EMatrixCode.PrintGrowth

Normalized measured print growth value, as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.

[VB6]

PrintGrowth As Single

read-only

Remarks

The use of the [EMatrixCode::PrintGrowth](#) property is a bit tricky: first a raw measure of the print growth is provided as [EMatrixCode::MeasuredPrintGrowth](#). Then, the measurement is normalized from the [EMatrixCodeReader::MinimumPrintGrowth](#) / [EMatrixCodeReader::MaximumPrintGrowth](#) / [EMatrixCodeReader::NominalPrintGrowth](#) properties of the [EMatrixCodeReader](#) instance, which must be provided by the user. The normalized [EMatrixCode::PrintGrowth](#) ranges around **0**.

EMatrixCode.PrintGrowthGrade

Measured print growth grading (**4** to **0** scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.

[VB6]

PrintGrowthGrade As Long

read-only

Remarks

Read-only.

EMatrixCode.ReadingThreshold

Absolute threshold (**0** to **255** scale) that was used during reading of the symbol.

[VB6]

ReadingThreshold As Long

read-only

Remarks

Read-only.

EMatrixCode.Save

```
[VB6]  
  
void Save(  
    ESerializer serializer  
)  
  
void Save(  
    String stream  
)
```

Parameters

serializer

-

stream

-

EMatrixCode.SemiT10GradingParameters

Semi T10-0701 grading parameters

```
[VB6]
```

SemiT10GradingParameters As EMatrixCodeSemiT10GradingParameters

read-only

Remarks

Read-only.

EMatrixCode.SetCorner

Sets a matrix code corner.

[VB6]

```
void SetCorner(  
    Long index,  
    EPoint corner  
)
```

Parameters

index

Index of the matrix code corner.

corner

New corner.

EMatrixCode.SymbolContrastSNR

Measured symbol contrast signal to noise ratio value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.

[VB6]

```
SymbolContrastSNR As Single
```

read-only

Remarks

Read-only.

EMatrixCode.UnusedErrorCorrection

Measured unused error correction value (**1.0** to **0** scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.

[VB6]

UnusedErrorCorrection As Single

read-only

Remarks

The **EMatrixCode::UnusedErrorCorrection** property takes into account the number of redundant bits used for error correction only; no erasure nor error detection bits are considered.

EMatrixCode.UnusedErrorCorrectionGrade

Measured unused error correction grading (**4** to **0** scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.

[VB6]

UnusedErrorCorrectionGrade As Long

read-only

Remarks

Read-only.

EMatrixCode.VerticalMarkGrowth

Measured vertical mark growth value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.

[VB6]

VerticalMarkGrowth As Single

read-only

Remarks

Read-only.

EMatrixCode.VerticalMarkMisplacement

Measured vertical mark misplacement value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.

[VB6]

VerticalMarkMisplacement As Single

read-only

Remarks

Read-only.

EMatrixCodeReader Class

A **EMatrixCodeReader** instance is a tool that processes an ROI and returns a **EMatrixCode** instance.

Remarks

PROPERTIES

ComputeGrading

Allows to choose whether the grading properties of the **EMatrixCode** object will be computed by the **EMatrixCodeReader::Reset**, **EMatrixCodeReader::Read** or **EMatrixCodeReader::LearnMore** methods.

MaxHeightWidthRatio

Maximum value for a Data Matrix aspect ratio

MaximumPrintGrowth	Maximum reference value in use for normalization of the PrintGrowth quality indicator.
MinimumPrintGrowth	Minimum reference value in use for normalization of the PrintGrowth quality indicator.
NominalPrintGrowth	Nominal reference value in use for normalization of the PrintGrowth quality indicator.
SearchParams	Parameter space that the algorithm uses to read a Data Matrix code from an ROI.
TimeOut	M Time-out for the EMatrixCodeReader::Learn , EMatrixCodeReader::LearnMore and R ead E methods.
THODS	
EMatrixCodeReader	Default constructor for EMATRIXCODEREADER objects.
GetLearnMaskElement	Allows to know which decoded parameters are learnt when the EMatrixCodeReader::Learn or EMatrixCodeReader::LearnMore methods are called.
Learn	Tries to locate, decode and read the Data Matrix code in the given ROI.

LearnMore	Tries to locate, decode and read the Data Matrix code in the given ROI.
Load	-
Read	Tries to locate, decode and read the Data Matrix code in the given ROI.
Reset	Resets the parameter search space to its default: the full range of all parameters.
Save	-
SetIso29158CalibrationParameters	Set ISO/IEC 29158 calibration paramters
SetLearnMaskElement	Allows to choose which decoded parameters are learnt when the EMatrixCodeReader::Learn or EMatrixCodeReader::LearnMore methods are called.

atrixCodeReader.ComputeGrading

Allows to choose whether the grading properties of the [EMatrixCode](#) object will be computed by the [EMatrixCodeReader::Reset](#), [EMatrixCodeReader::Read](#) or [EMatrixCodeReader::LearnMore](#) methods.

[VB6]

ComputeGrading As Boolean

read-write

Remarks

Default: **FALSE**.

EMatrixCodeReader.EMatrixCodeReader

Default constructor for EMatrixCodeReader objects.

[VB6]

```
void EMatrixCodeReader()
)
void EMatrixCodeReader(
    EMatrixCodeReader other
)
```

Parameters

other

EMatrixCodeReader.GetLearnMaskElement

Allows to know which decoded parameters are learnt when the [EMatrixCodeReader::Learn](#) or [EMatrixCodeReader::LearnMore](#) methods are called.

[VB6]

```
Boolean GetLearnMaskElement(
    ELearnParam index
)
```

Parameters*index*Parameter identifier, as defined in [ELearnParam](#)

EMatrixCodeReader.Learn

Tries to locate, decode and read the Data Matrix code in the given ROI.

[VB6]

```
EMatrixCode Learn(
    EROIBW8 roi
)
```

Parameters*roi*

ROI in which the Data Matrix has to be found.

Remarks

If successful, it adds the parameters of the Data Matrix code found into the internal learning database. The decoding results can be found in the returned [EMatrixCode](#) object. The addition of the parameters of the Data Matrix code found into the internal learning database means that subsequent **Read** operations will be faster, but can only be performed on similar Data Matrix codes/conditions. Only the parameters that are tagged for learning are remembered for the subsequent [EMatrixCodeReader::Read](#) operations (see [EMatrixCodeReader::SetLearnMaskElement](#)). See the [EMatrixCode::Found](#) property for information about the outcome of the **Learn** process.

EMatrixCodeReader.LearnMore

Tries to locate, decode and read the Data Matrix code in the given ROI.

[VB6]

```
EMatrixCode LearnMore(
    EROIBW8 roi
)
```

Parameters

roi

ROI in which the Data Matrix has to be found.

Remarks

If successful, it cumulates the parameters of the Data Matrix code found with those already present in the internal learning database. The decoding results can be found in the returned **EMatrixCode** object. The cumulation of the parameters of the Data Matrix code found with those already present in the internal learning database means that subsequent **Read** operations will be faster, but can only be performed on similar Data Matrix codes/conditions. Only the parameters that are tagged for learning are remembered for the subsequent **EMatrixCodeReader::Read** operations (see **EMatrixCodeReader::SetLearnMaskElement**). See the **EMatrixCode::Found** property for information about the outcome of the **LearnMore** process.

EMatrixCodeReader.Load

[VB6]

```
void Load(
    ESerializer serializer
)

void Load(
    String stream
)
```

Parameters

serializer

-

stream

EMatrixCodeReader.MaxHeightWidthRatio

Maximum value for a Data Matrix aspect ratio

[VB6]

MaxHeightWidthRatio As Single

read-write

Remarks

This property allows controlling what kind of objects are considered as potential MatrixCode instances in the image. When objects are found in the image, only those where the bounding box has an aspect ratio smaller than this value are taken into account for digitization and decoding. The default value is 3.8, and should be adjusted if the MatrixCode cells in your image are non-square, or if your matrix code uses a very non-square symbology such as 32x8. The supplied value must lie between 0.0 and 5.0.

EMatrixCodeReader.MaximumPrintGrowth

Maximum reference value in use for normalization of the **PrintGrowth** quality indicator.

[VB6]

MaximumPrintGrowth As Single

read-write

Remarks

Default: **2.0**. After the standard, parameter **PrintGrowth** must be computed as normalized value related to three references: minimum, nominal and maximum values have to be provided. Parameter **MeasuredPrintGrowth** is the raw, un-normalized print quality parameter. It is computed as the measured area of the active cells over the area of ideal square cells, having sides equal to the pitches (such cells perfectly tile the symbol). Its value typically is on the order of unity. The **PrintGrowth** is derived from the **MeasuredPrintGrowth** by means of the three normalization parameters.

EMatrixCodeReader.MinimumPrintGrowth

Minimum reference value in use for normalization of the **PrintGrowth** quality indicator.

[VB6]

MinimumPrintGrowth As Single

read-write

Remarks

Default: **0.0**. After the standard, parameter **PrintGrowth** must be computed as normalized value related to three references: minimum, nominal and maximum values have to be provided. Parameter **MeasuredPrintGrowth** is the raw, un-normalized print quality parameter. It is computed as the measured area of the active cells over the area of ideal square cells, having sides equal to the pitches (such cells perfectly tile the symbol). Its value typically is on the order of unity. The **PrintGrowth** is derived from the **MeasuredPrintGrowth** by means of the three normalization parameters.

EMatrixCodeReader.NominalPrintGrowth

Nominal reference value in use for normalization of the **PrintGrowth** quality indicator.

[VB6]

NominalPrintGrowth As Single

read-write

Remarks

Default: **1.0**. After the standard, parameter **PrintGrowth** must be computed as normalized value related to three references: minimum, nominal and maximum values have to be provided. Parameter **MeasuredPrintGrowth** is the raw, un-normalized print quality parameter. It is computed as the measured area of the active cells over the area of ideal square cells, having sides equal to the pitches (such cells perfectly tile the symbol). Its value typically is on the order of unity. The **PrintGrowth** is derived from the **MeasuredPrintGrowth** by means of the three normalization parameters.

EMatrixCodeReader.Read

Tries to locate, decode and read the Data Matrix code in the given ROI.

```
[VB6]
EMatrixCode Read(
    EROIBW8 roi
)
```

Parameters

roi

ROI in which the Data Matrix has to be found.

Remarks

The decoding results can be found in the returned [EMatrixCode](#) object. See the [EMatrixCode::Found](#) property for information about the outcome of the **Read** process.

Note. This function throws an exception if the matrix code in the given ROI can not be read.

EMatrixCodeReader.Reset

Resets the parameter search space to its default: the full range of all parameters.

```
[VB6]  
void Reset()  
)
```

Remarks

This does not modify the learning mask.

EMatrixCodeReader.Save

-

```
[VB6]  
void Save(  
    ESerializer serializer  
)  
  
void Save(  
    String stream  
)
```

Parameters

serializer

-

stream

-

EMatrixCodeReaderSearchParams

Parameter space that the algorithm uses to read a Data Matrix code from an ROI.

[VB6]

SearchParams As ESearchParamsType

read-only

Remarks

It can be modified through automatic learning (using the [EMatrixCodeReader::Learn](#) or [EMatrixCodeReader::LearnMore](#) methods) or by using [EMatrixCodeReader::SearchParams](#) properties and methods.

EMatrixCodeReader.SetIso29158CalibrationParameters

Set ISO/IEC 29158 calibration parameters

[VB6]

```
void SetIso29158CalibrationParameters(
    Single Rcal,
    Single MLcal,
    Single SRcal,
    Single SRtarget
)
```

Parameters

Rcal

-

MLcal

-

SRcal

-

SRtarget

-

EMatrixCodeReader.SetLearnMaskElement

Allows to choose which decoded parameters are learnt when the [EMatrixCodeReader::Learn](#) or [EMatrixCodeReader::LearnMore](#) methods are called.

[VB6]

```
void SetLearnMaskElement(  
    ELearnParam index,  
    Boolean value  
)
```

Parameters

index

Parameter identifier, as defined in [ELearnParam](#).

value

TRUE to enable the parameter for learning.

Remarks

In order to enable a parameter for learning, you need to set corresponding item of LearnMask to TRUE. Default: all items are set to TRUE.

EMatrixCodeReader.TimeOut

Time-out for the [EMatrixCodeReader::Learn](#), [EMatrixCodeReader::LearnMore](#) and [Read](#) methods.

[VB6]

```
TimeOut As Long
```

read-write

Remarks

If the processing time of one of these functions becomes longer than the set time-out, the processing is stopped and an exception is thrown. In that case, the error code of the exception is [EError_TimeoutReached](#). The time-out is set in microseconds. This time-out is not a real time-out. The processing is stopped as soon as possible after the time-out has been reached. This means that the time elapsed effectively in the method can be greater than the time-out in itself.

EMeasurementUnit Class

The measurement units that are supported by Open eVision.

Remarks

Measurement units are used to represent physical units, such as "meter" or "inch", and ease conversions between different unit systems. They are used to build dimensional values.

PROPERTIES

Magnitude

Relative magnitude of this unit, with respect to a standard unit (e.g. 1 mm = 0.001 m).

Name

METHODS

Pointer to a **NULL**-terminated string containing the unit abbreviation.

ConversionFactorTo

Returns the factor needed to convert from this measurement unit to a second one.

EMeasurementUnit

Constructs a measurement unit.

GetStockMeasurementUnit

E-

M

easurementUnit.ConversionFactorTo

Returns the factor needed to convert from this measurement unit to a second one.

[VB6]

```
Single ConversionFactorTo(
    EMeasurementUnit Unit
)
```

Parameters

Unit

Reference to the second measurement unit.

EMeasurementUnit.EMeasurementUnit

Constructs a measurement unit.

[VB6]

```
void EMeasurementUnit(
    Single magnitude,
    String name
)

void EMeasurementUnit(
    EMeasurementUnit pUnit
)

void EMeasurementUnit(
)
```

Parameters

magnitude

Relative magnitude of this unit with respect to a standard (e.g. 1 mm = 0.001 m).

name

Unit abbreviation (e.g. "**mm**").

pUnit

EMeasurementUnit.GetStockMeasurementUnit

-

[VB6]

```
EMeasurementUnit GetStockMeasurementUnit(  
    EStockMeasurementUnit unit  
)
```

Parameters

unit

EMeasurementUnit.Magnitude

Relative magnitude of this unit, with respect to a standard unit (e.g. 1 mm = 0.001 m).

[VB6]

Magnitude As Single

read-write

EMeasurementUnit.Name

Pointer to a **NULL**-terminated string containing the unit abbreviation.

[VB6]

Name As String

read-write

EMovingAverage Class

Temporal integration of a number of images to reduce noise.

PROPERTIES

SrcImage

M Returns the image in which you must store the next image to be averaged.

E

THODS

Average

Performs averaging of the source image with the preceding ones, and computes the de-noised image.

EMovingAverage

Constructs an EMovingAverage object.

GetSize

Queries a moving average context for the current parameter settings.

Reset

Restarts the average process as if no image had ever been handed to the EMovingAverage object.

SetSize

EInitializes a moving average context with appropriate parameters.

M

ovingAverage.Average

Performs averaging of the source image with the preceding ones, and computes the de-noised image.

[VB6]

```
void Average(
    EROIBW8 destinationImage
)
void Average(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage
)
```

Parameters

destinationImage

Pointer to the destination image.

sourceImage

Pointer to the source image.

Remarks

The overload with only the destinationImage may be used only when the buffer allocation scheme has been set to internal (see [EMovingAverage::SetSize](#) or [EMovingAverage::EMovingAverage](#))

EMovingAverage.EMovingAverage

Constructs an EMovingAverage object.

[VB6]

```
void EMovingAverage(
    EMovingAverage other
)
void EMovingAverage(
)
void EMovingAverage(
    Long period,
    Long width,
    Long height,
    Boolean internalAllocationScheme
)
```

Parameters

other

-

period

Number of images on which to integrate. A power of 2 is recommended.

width

Image width (all images used for averaging must be of the same size).

height

Image height (all images used for averaging must be of the same size).

internalAllocationScheme

Buffer allocation scheme. When **TRUE**, the moving average context provides the image to be acquired into (see member [EMovingAverage::SrclImage](#)).

Remarks

The default constructor constructs a void moving average context. A void moving average context has no internal buffers allocated and cannot be used for integration. Use the [EMovingAverage::SetSize](#) member after construction, or the initializing constructor instead. The sizing constructor constructs and initializes a moving average context.

EMovingAverage.GetSize

Queries a moving average context for the current parameter settings.

```
[VB6]  
void GetSize(  
    Long numberOfImages,  
    Long imageWidth,  
    Long imageHeight,  
    Boolean internalAllocationScheme  
)
```

Parameters

numberOfImages

Number of images on which to integrate.

imageWidth

Image width (all images used for averaging must be of the same size).

imageHeight

Image height (all images used for averaging must be of the same size).

internalAllocationScheme

Buffer allocation scheme. When **TRUE**, the moving average context provides the image to be acquired into (see member [EMovingAverage::SrclImage](#)).

EMovingAverage.Reset

Restarts the average process as if no image had ever been handed to the EMovingAverage object.

```
[VB6]  
void Reset(  
)
```

Remarks

The behavior is thus the same as after a [EMovingAverage::SetSize](#) operation.

EMovingAverage.SetSize

Initializes a moving average context with appropriate parameters.

[VB6]

```
void SetSize(  
    Long numberOfImages,  
    Long imageWidth,  
    Long imageHeight,  
    Boolean internalAllocationScheme  
)
```

Parameters

numberOfImages

Number of images on which to integrate. A power of 2 is recommended.

imageWidth

Image width.

imageHeight

Image height.

internalAllocationScheme

Buffer allocation scheme. When **TRUE**, the moving average context provides the image to be acquired into (see member [EMovingAverage::SrcImage](#)).

EMovingAverage.SrcImage

Returns the image in which you must store the next image to be averaged.

[VB6]

```
SrcImage As EImageBW8
```

read-only

Remarks

This method may be used only when the buffer allocation scheme has been set to internal (see [EMovingAverage::SetSize](#) or [EMovingAverage::EMovingAverage](#))

EObject Class

This class represents an object (blob) in an encoded image.

Remarks

This class inherits from the ECodedElement class and provides additional methods to access the holes of a particular object.

The extraction of the holes is lazy. This means that the holes are not computed before they get accessed. For this reason, the first access to the holes is slower than the subsequent accesses. On the other hand, the applications that do not make use of the holes are not penalized by the cost of hole extraction.

Base Class: [ECodedElement](#)

PROPERTIES

HoleCount

M

Returns the number of holes in the object.

THODS

GetHole

E

Returns a specified hole in the object.

EObject.GetHole

Returns a specified hole in the object.

[VB6]

```
EHole GetHole(  
    Long index  
)
```

Parameters

index

The index of the hole of interest.

EObject.HoleCount

Returns the number of holes in the object.

[VB6]

```
HoleCount As Long  
read-only
```

EObjectRunsIterator Class

Iterator to the runs of a coded element.

Remarks

This class is responsible for the sequential access to the individual runs of a coded element. A run is defined as a maximal sequence of consecutive pixels on the same run, that all belong to the same coded element.

PROPERTIES**EndX**

Returns the abscissa (X-coordinate) of the rightmost pixel of the run the iterator is currently over.

IsDone

Tests whether all the runs have been spanned.

Length

Returns the lengths of the run the iterator is currently over.

StartX

Returns the abscissa (X-coordinate) of the leftmost pixel of the run the iterator is currently over.

Y

M Returns the ordinate (Y-coordinate) of the run the iterator is currently over.

E**THODS****EObjectRunsIterator**

Constructs an iterator to the runs of a coded element.

First

Rewinds to the first run of the coded element.

Next

Advance to the next run in the iterator.

operator=

|
E Assignment operator.

O

bjectRunsIterator.EndX

Returns the abscissa (X-coordinate) of the rightmost pixel of the run the iterator is currently over.

[VB6]

EndX As Long

read-only

Remarks

An exception is thrown if the iterator has reached its end. Use [EOBJECTRUNSITERATOR::ISDONE](#) to check this condition.

EOBJECTRUNSITERATOR.EOBJECTRUNSITERATOR

Constructs an iterator to the runs of a coded element.

[VB6]

```
void EObjectRunsIterator()  
)  
  
void EObjectRunsIterator(  
ECodedElement codedElement  
)  
  
void EObjectRunsIterator(  
EOBJECTRUNSITERATOR other  
)
```

Parameters

codedElement

The coded element of interest, in the case the iterator is to be constructed from a given coded element.

other

The iterator to be copied, in the case of the copy constructor.

EObjectRunsIterator.First

Rewinds to the first run of the coded element.

[VB6]

```
void First()  
)
```

EObjectRunsIterator.IsDone

Tests whether all the runs have been spanned.

[VB6]

```
IsDone As Boolean  
read-only
```

EObjectRunsIterator.Length

Returns the lengths of the run the iterator is currently over.

[VB6]

Length As Long

read-only

Remarks

An exception is thrown if the iterator has reached its end. Use [EObjectRunsIterator::IsDone](#) to check this condition.

EObjectRunsIterator.Next

Advance to the next run in the iterator.

[VB6]

```
void Next()  
)
```

Remarks

An exception is thrown if the iterator has reached its end. Use [EObjectRunsIterator::IsDone](#) to check this condition.

EObjectRunsIterator.operator=

Assignment operator.

[VB6]

```
EObjectRunsIterator operator=(  
    EObjectRunsIterator other  
)
```

Parameters

other

The iterator to be copied.

EObjectRunsIterator.StartX

Returns the abscissa (X-coordinate) of the leftmost pixel of the run the iterator is currently over.

[VB6]

StartX As Long

read-only

Remarks

An exception is thrown if the iterator has reached its end. Use [EObjectRunsIterator::IsDone](#) to check this condition.

EObjectRunsIterator.Y

Returns the ordinate (Y-coordinate) of the run the iterator is currently over.

[VB6]

Y As Long

read-only

Remarks

An exception is thrown if the iterator has reached its end. Use [EObjectRunsIterator::IsDone](#) to check this condition.

EObjectSelection Class

This container class handles the selection of a subset of coded elements taken from a coded image.

Remarks

This class provides methods to perform a selection of objects or holes and to retrieve the features of the coded elements in the collection.

PROPERTIES

AttachedImage

Sets the attached image for computing the features that depend on an image.

ElementCount

Returns the number of coded elements selection.

FeretAngle

M Angle of the Feret box (in the current angle units).

E

THODS

Add

Add a coded element to the selection.

AddHole

Adds a single hole of a coded image.

AddHoles

Adds all the holes contained in an object, in a layer or in a coded image.

AddHolesOfSelectedObjects	Adds the holes of all the objects that are currently selected.
AddLayer	Adds all the objects of a single layer in a coded image.
AddObject	Adds a single object of a layer in a coded image.
AddObjects	Adds all the objects of all the layers of a given coded image.
AddObjectsUsingFloatFeature	Selects the objects that fullfil a condition on a floating-point feature.
AddObjectsUsingIntegerFeature	Selects the objects that fullfil a condition on an integer feature.
AddObjectsUsingRectangle	Adds all the objects of a coded image whose location fullfil a criterion based on a rectangle.
AddObjectsUsingUnsignedIntegerFeature	Selects the objects that fullfil a condition on an unsigned integer feature.
AddObjectUsingPosition	Adds the object of a coded image that lies at a particular location.
Clear	Remove all the coded elements that are contained in the selection.

ClearFeatureCache	Clears the internal cache for the computed features.
EObjectSelection	-
FeatureAverage	Computes the average of the values of the given feature across the selection.
FeatureDeviation	Computes the standard deviation of the values of the given feature across the selection.
FeatureVariance	Computes the variance of the values of the given feature across the selection.
FloatFeatureMaximum	Computes the maximum value of the given feature across the selection.
FloatFeatureMinimum	Computes the minimum value of the given feature across the selection.
GetElement	Index-based access to the coded elements of the selection.
GetFloatFeature	Returns the value of a floating-point feature for a selected coded element.

[GetIndexOfElement](#)

Retrieves the index of a given coded element in the selection.

[GetIntegerFeature](#)

Returns the value of an integer feature for a selected coded element.

[GetUnsignedIntegerFeature](#)

Returns the value of an unsigned integer feature for a selected coded element.

[IntegerFeatureMaximum](#)

Computes the maximum value of the given feature across the selection.

[IntegerFeatureMinimum](#)

Computes the minimum value of the given feature across the selection.

[IsSelected](#)

Tests whether a given coded element is present in the selection.

[Remove](#)

Remove a coded element from the selection.

[RemoveHole](#)

Removes a single hole of a coded image.

[RemoveHoles](#)

Removes all the holes contained in an object, in a layer or in a coded image.

[RemoveLayer](#)

Removes all the objects of a single layer in a coded image.

[RemoveObject](#)

Removes a single object of a layer in a coded image.

[RemoveObjectsUsingRectangle](#)

Removes all the objects of a coded image whose location fullfil a criterion based on a rectangle.

[RemoveObjectUsingPosition](#)

Removes the object of a coded image that lies at a particular location.

[RemoveSelectedHoles](#)

Remove all the holes that are currently present in the selection.

[RemoveUsingFloatFeature](#)

Removes the selected coded elements that fullfil a condition on a floating-point feature.

[RemoveUsingIntegerFeature](#)

Removes the selected coded elements that fullfil a condition on an unsigned integer feature.

[RemoveUsingUnsignedIntegerFeature](#)

Removes the selected coded elements that fullfil a condition on an unsigned integer feature.

[RenderMask](#)

Creates a Flexible Mask from the selection.

Sort

Sorts the selected coded elements according to the value of a feature.

[UnsignedIntegerFeatureMaximum](#)

Computes the maximum value of the given feature across the selection.

[UnsignedIntegerFeatureMinimum](#)

Computes the minimum value of the given feature across the selection.

objectSelection.Add

Add a coded element to the selection.

[VB6]

```
void Add(  
    ECodedElement element  
)
```

Parameters

element

The coded element.

EObjectSelection.AddHole

Adds a single hole of a coded image.

[VB6]

```

void AddHole(
    ECodedImage2 codedImage,
    Long objectIndex,
    Long holeIndex
)

void AddHole(
    ECodedImage2 codedImage,
    Long layerIndex,
    Long objectIndex,
    Long holeIndex
)

```

Parameters

codedImage

The coded image.

objectIndex

The index of the parent object in the layer.

holeIndex

The index of the hole in the parent object.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

Remarks

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

EObjectSelection.AddHoles

Adds all the holes contained in an object, in a layer or in a coded image.

[VB6]

```

void AddHoles(
    ECodedImage2 codedImage
)

```

```
void AddHoles(
    ECodedImage2 codedImage,
    Long layerIndex
)

void AddHoles(
    ECodedImage2 codedImage,
    Long layerIndex,
    Long objectIndex
)
```

Parameters

codedImage

The source coded image.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, all the layers are taken into consideration.

objectIndex

The index of the parent object. If this parameter is left unspecified, all the objects are taken into consideration.

EObjectSelection.AddHolesOfSelectedObjects

Adds the holes of all the objects that are currently selected.

[VB6]

```
void AddHolesOfSelectedObjects(
)
```

EObjectSelection.AddLayer

Adds all the objects of a single layer in a coded image.

```
[VB6]

void AddLayer(
    ECodedImage2 codedImage,
    Long layerIndex
)

void AddLayer(
    ECodedImage2 codedImage
)
```

Parameters

codedImage

The coded image.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

Remarks

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

EObjectSelection.AddObject

Adds a single object of a layer in a coded image.

```
[VB6]

void AddObject(
    ECodedImage2 codedImage,
    Long objectIndex
)

void AddObject(
    ECodedImage2 codedImage,
    Long layerIndex,
    Long objectIndex
)
```

Parameters

codedImage

The coded image.

objectIndex

The index of the object in the layer.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

Remarks

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

EObjectSelection.AddObjects

Adds all the objects of all the layers of a given coded image.

[VB6]

```
void AddObjects(  
    ECodedImage2 image  
)
```

Parameters

image

The coded image.

EObjectSelection.AddObjectsUsingFloatFeature

Selects the objects that fullfil a condition on a floating-point feature.

[VB6]

```
void AddObjectsUsingFloatFeature(
    ECodedImage2 codedImage,
    Long layerIndex,
    EFeature feature,
    Single threshold,
    ESingleThresholdMode mode
)

void AddObjectsUsingFloatFeature(
    ECodedImage2 codedImage,
    Long layerIndex,
    EFeature feature,
    Single lowBound,
    Single highBound,
    EDoubleThresholdMode mode
)

void AddObjectsUsingFloatFeature(
    ECodedImage2 codedImage,
    EFeature feature,
    Single threshold,
    ESingleThresholdMode mode
)

void AddObjectsUsingFloatFeature(
    ECodedImage2 codedImage,
    EFeature feature,
    Single lowBound,
    Single highBound,
    EDoubleThresholdMode mode
)
```

Parameters

codedImage

The source coded image.

layerIndex

The index of the layer of interest. If left unspecified, all the layers are taken into consideration.

feature

The feature that serves as a filter.

threshold

The single threshold on the feature.

mode

Specifies the way the threshold is interpreted.

lowBound

The low bound of the range on the feature.

highBound

The high bound of the range on the feature.

Remarks

Most features are floating-point. These methods thus tend to be the most widely used.

EObjectSelection.AddObjectsUsingIntegerFeature

Selects the objects that fulfil a condition on an integer feature.

[VB6]

```
void AddObjectsUsingIntegerFeature(
    ECodedImage2 codedImage,
    Long layerIndex,
    EFeature feature,
    Long threshold,
    ESingleThresholdMode mode
)

void AddObjectsUsingIntegerFeature(
    ECodedImage2 codedImage,
    Long layerIndex,
    EFeature feature,
    Long lowBound,
    Long highBound,
    EDoubleThresholdMode mode
)

void AddObjectsUsingIntegerFeature(
    ECodedImage2 codedImage,
    EFeature feature,
    Long threshold,
    ESingleThresholdMode mode
)

void AddObjectsUsingIntegerFeature(
    ECodedImage2 codedImage,
    EFeature feature,
    Long lowBound,
    Long highBound,
    EDoubleThresholdMode mode
)
```

Parameters

codedImage

The source coded image.

layerIndex

The index of the layer of interest. If left unspecified, all the layers are taken into consideration.

feature

The feature that serves as a filter.

threshold

The single threshold on the feature.

mode

Specifies the way the threshold is interpreted.

lowBound

The low bound of the range on the feature.

highBound

The high bound of the range on the feature.

EObjectSelection.AddObjectsUsingRectangle

Adds all the objects of a coded image whose location fullfil a criterion based on a rectangle.

[VB6]

```
void AddObjectsUsingRectangle(
    ECodedImage2 codedImage,
    Long x,
    Long y,
    Long width,
    Long height,
    ERectangleMode mode
)
```

```
void AddObjectsUsingRectangle(
    ECodedImage2 codedImage,
    Long layerIndex,
    Long x,
    Long y,
    Long width,
    Long height,
    ERectangleMode mode
)
```

Parameters

codedImage

The source coded image.

x

The X-coordinate of the top-left corner of the selection rectangle.

y

The Y-coordinate of the top-left corner of the selection rectangle.

width

The width of the selection rectangle.

height

The height of the selection rectangle.

mode

The comparison mode with respect to the selection rectangle.

layerIndex

If specified, only the specified layer is taken into consideration.

EObjectSelection.AddObjectsUsingUnsignedIntegerFeature

Selects the objects that fullfil a condition on an unsigned integer feature.

[VB6]

```
void AddObjectsUsingUnsignedIntegerFeature(
    ECodedImage2 codedImage,
    Long layerIndex,
    EFeature feature,
    Long threshold,
    ESingleThresholdMode mode
)

void AddObjectsUsingUnsignedIntegerFeature(
    ECodedImage2 codedImage,
    EFeature feature,
    Long threshold,
    ESingleThresholdMode mode
)

void AddObjectsUsingUnsignedIntegerFeature(
    ECodedImage2 codedImage,
    Long layerIndex,
    EFeature feature,
    Long lowBound,
    Long highBound,
    EDoubleThresholdMode mode
)

void AddObjectsUsingUnsignedIntegerFeature(
    ECodedImage2 codedImage,
    EFeature feature,
    Long lowBound,
    Long highBound,
    EDoubleThresholdMode mode
)
```

Parameters

codedImage

The source coded image.

layerIndex

The index of the layer of interest. If left unspecified, all the layers are taken into consideration.

feature

The feature that serves as a filter.

threshold

The single threshold on the feature.

mode

Specifies the way the threshold is interpreted.

lowBound

The low bound of the range on the feature.

highBound

The high bound of the range on the feature.

EObjectSelection.AddObjectUsingPosition

Adds the object of a coded image that lies at a particular location.

[VB6]

```
void AddObjectUsingPosition(
    ECodedImage2 codedImage,
    Long x,
    Long y
)
```

Parameters

codedImage

The source coded image.

x

The X-coordinate of the object.

y

The Y-coordinate of the object.

Remarks

If no object lies at the specified coordinate, the selection is not changed.

EObjectSelection.AttachedImage

Sets the attached image for computing the features that depend on an image.

[VB6]

AttachedImage As EROIBW8

read-write

Remarks

An image must be attached before dealing with the following features: [EFeature_PixelMin](#), [EFeature_PixelMax](#), [EFeature_WeightedGravityCenterX](#), [EFeature_WeightedGravityCenterY](#), [EFeature_PixelGrayAverage](#), [EFeature_PixelGrayVariance](#), [EFeature_PixelGrayDeviation](#).

EObjectSelection.Clear

Remove all the coded elements that are contained in the selection.

[VB6]
void Clear(
)

EObjectSelection.ClearFeatureCache

Clears the internal cache for the computed features.

[VB6]
void ClearFeatureCache(
)

Remarks

This is useful to reduce memory consumption.

EObjectSelection.ElementCount

Returns the number of coded elements selection.

[VB6]

ElementCount As Long

read-only

EObjectSelection.EObjectSelection

-

[VB6]

```
void EObjectSelection(  
    EObjectSelection other  
)  
  
void EObjectSelection(  
)
```

Parameters

other

EObjectSelection.FeatureAverage

Computes the average of the values of the given feature across the selection.

[VB6]

```
Single FeatureAverage(
    EFeature feature
)
```

Parameters

feature

The feature of interest.

EObjectSelection.FeatureDeviation

Computes the standard deviation of the values of the given feature across the selection.

[VB6]

```
Single FeatureDeviation(
    EFeature feature
)
```

Parameters

feature

The feature of interest.

EObjectSelection.FeatureVariance

Computes the variance of the values of the given feature across the selection.

[VB6]

```
Single FeatureVariance(
    EFeature feature
)
```

Parameters

feature

The feature of interest.

EObjectSelection.FeretAngle

Angle of the Feret box (in the current angle units).

[VB6]

FeretAngle As Single

read-write

Remarks

A Feret angle must be set for the following features: [EFeature_FeretBoxCenterX](#), [EFeature_FeretBoxCenterY](#), [EFeature_FeretBoxWidth](#), [EFeature_FeretBoxHeight](#).

EObjectSelection.FloatFeatureMaximum

Computes the maximum value of the given feature across the selection.

[VB6]

```
Single FloatFeatureMaximum(
    EFeature feature
)
```

Parameters

feature

The feature of interest.

Remarks

Most features are floating-point. This method thus tends to be the most widely used.

EObjectSelection.FloatFeatureMinimum

Computes the minimum value of the given feature across the selection.

```
[VB6]  
Single FloatFeatureMinimum(  
    EFeature feature  
)
```

Parameters

feature

The feature of interest.

Remarks

Most features are floating-point. This method thus tends to be the most widely used.

EObjectSelection.GetElement

Index-based access to the coded elements of the selection.

```
[VB6]  
ECodedElement GetElement(  
    Long index  
)
```

Parameters

index

The index of the coded element of interest.

EObjectSelection.GetFloatFeature

Returns the value of a floating-point feature for a selected coded element.

[VB6]

```
Single GetFloatFeature(  
    Long index,  
    EFeature feature  
)
```

Parameters

index

The index of the selected coded element.

feature

The feature of interest.

Remarks

Most features are floating-point. This method thus tends to be the most widely used.

EObjectSelection.GetIndexOfElement

Retrieves the index of a given coded element in the selection.

[VB6]

```
Long GetIndexOfElement(  
    ECodedElement element  
)
```

Parameters

element

The coded element of interest.

Remarks

An exception is thrown if the coded element is not present in the selection.

EObjectSelection.GetIntegerFeature

Returns the value of an integer feature for a selected coded element.

[VB6]

```
Long GetIntegerFeature(  
    Long index,  
    EFeature feature  
)
```

Parameters

index

The index of the selected coded element.

feature

The feature of interest.

EObjectSelection.GetUnsignedIntegerFeature

Returns the value of an unsigned integer feature for a selected coded element.

[VB6]

```
Long GetUnsignedIntegerFeature(  
    Long index,  
    EFeature feature  
)
```

Parameters

index

The index of the selected coded element.

feature

The feature of interest.

EObjectSelection.IntegerFeatureMaximum

Computes the maximum value of the given feature across the selection.

[VB6]

```
Long IntegerFeatureMaximum(
    EFeature feature
)
```

Parameters

feature

The feature of interest.

EObjectSelection.IntegerFeatureMinimum

Computes the minimum value of the given feature across the selection.

[VB6]

```
Long IntegerFeatureMinimum(
    EFeature feature
)
```

Parameters

feature

The feature of interest.

EObjectSelection.isSelected

Tests whether a given coded element is present in the selection.

[VB6]

```

Boolean IsSelected(
    ECodedElement element
)

Boolean IsSelected(
    ECodedImage2 codedImage,
    Long objectIndex
)

Boolean IsSelected(
    ECodedImage2 codedImage,
    Long layerIndex,
    Long objectIndex
)

Boolean IsSelected(
    ECodedImage2 codedImage,
    Long layerIndex,
    Long objectIndex,
    Long holeIndex
)

```

Parameters

element

The coded element.

codedImage

The coded image.

objectIndex

The index of the object in the layer of interest.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

holeIndex

If specified, the index of the hole in the object. If unspecified, one tests the presence of the object.

Remarks

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

EObjectSelection.Remove

Remove a coded element from the selection.

```
[VB6]  
void Remove(  
    ECodedElement element  
)
```

Parameters

element
The coded element.

EObjectSelection.RemoveHole

Removes a single hole of a coded image.

```
[VB6]  
void RemoveHole(  
    ECodedImage2 codedImage,  
    Long objectIndex,  
    Long holeIndex  
)  
  
void RemoveHole(  
    ECodedImage2 codedImage,  
    Long layerIndex,  
    Long objectIndex,  
    Long holeIndex  
)
```

Parameters

codedImage
The coded image.
objectIndex

The index of the parent object in the layer.

holeIndex

The index of the hole in the parent object.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

Remarks

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

EObjectSelection.RemoveHoles

Removes all the holes contained in an object, in a layer or in a coded image.

```
[VB6]
void RemoveHoles(
    ECodedImage2 codedImage
)
void RemoveHoles(
    ECodedImage2 codedImage,
    Long layerIndex
)
void RemoveHoles(
    ECodedImage2 codedImage,
    Long layerIndex,
    Long objectIndex
)
```

Parameters

codedImage

The source coded image.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, all the layers are taken into consideration.

objectIndex

The index of the parent object. If this parameter is left unspecified, all the objects are taken into consideration.

EObjectSelection.RemoveLayer

Removes all the objects of a single layer in a coded image.

```
[VB6]  
  
void RemoveLayer(  
    ECodedImage2 codedImage  
)  
  
void RemoveLayer(  
    ECodedImage2 codedImage,  
    Long layerIndex  
)
```

Parameters

codedImage

The coded image.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

Remarks

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

EObjectSelection.RemoveObject

Removes a single object of a layer in a coded image.

```
[VB6]  
  
void RemoveObject(  
    ECodedImage2 codedImage,  
    Long objectIndex  
)
```

```
void RemoveObject(
    ECodedImage2 codedImage,
    Long layerIndex,
    Long objectIndex
)
```

Parameters*codedImage*

The coded image.

objectIndex

The index of the object in the layer.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

Remarks

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

EObjectSelection.RemoveObjectsUsingRectangle

Removes all the objects of a coded image whose location fullfil a criterion based on a rectangle.

[VB6]

```
void RemoveObjectsUsingRectangle(
    ECodedImage2 codedImage,
    Long x,
    Long y,
    Long width,
    Long height,
    ERectangleMode mode
)
```

```
void RemoveObjectsUsingRectangle(
    ECodedImage2 codedImage,
    Long layerIndex,
    Long x,
    Long y,
    Long width,
    Long height,
    ERectangleMode mode
)
```

Parameters

codedImage

The source coded image.

x

The X-coordinate of the top-left corner of the selection rectangle.

y

The Y-coordinate of the top-left corner of the selection rectangle.

width

The width of the selection rectangle.

height

The height of the selection rectangle.

mode

The comparison mode with respect to the selection rectangle.

layerIndex

If specified, only the specified layer is taken into consideration.

EObjectSelection.RemoveObjectUsingPosition

Removes the object of a coded image that lies at a particular location.

[VB6]

```
void RemoveObjectUsingPosition(
    ECodedImage2 codedImage,
    Long x,
    Long y
)
```

Parameters

codedImage

The source coded image.

x

The X-coordinate of the object.

y

The Y-coordinate of the object.

Remarks

If no object lies at the specified coordinate, the selection is not changed.

EObjectSelection.RemoveSelectedHoles

Remove all the holes that are currently present in the selection.

[VB6]

```
void RemoveSelectedHoles(
)
```

EObjectSelection.RemoveUsingFloatFeature

Removes the selected coded elements that fullfil a condition on a floating-point feature.

[VB6]

```
void RemoveUsingFloatFeature(
    EFeature feature,
    Single threshold,
    ESingleThresholdMode mode
)

void RemoveUsingFloatFeature(
    EFeature feature,
    Single low,
    Single high,
    EDoubleThresholdMode mode
)
```

Parameters

feature

The feature that serves as a filter.

threshold

The single threshold on the feature.

mode

Specifies the way the threshold is interpreted.

low

The low bound of the range on the feature.

high

The high bound of the range on the feature.

Remarks

Most features are floating-point. These methods thus tend to be the most widely used.

EObjectSelection.RemoveUsingIntegerFeature

Removes the selected coded elements that fullfil a condition on an unsigned integer feature.

[VB6]

```
void RemoveUsingIntegerFeature(
    EFeature feature,
    Long threshold,
    ESingleThresholdMode mode
)

void RemoveUsingIntegerFeature(
    EFeature feature,
    Long low,
    Long high,
    EDoubleThresholdMode mode
)
```

Parameters

feature

The feature that serves as a filter.

threshold

The single threshold on the feature.

mode

Specifies the way the threshold is interpreted.

low

The low bound of the range on the feature.

high

The high bound of the range on the feature.

EObjectSelection.RemoveUsingUnsignedIntegerFeature

Removes the selected coded elements that fullfil a condition on an unsigned integer feature.

[VB6]

```
void RemoveUsingUnsignedIntegerFeature(
    EFeature feature,
    Long threshold,
    ESingleThresholdMode mode
)

void RemoveUsingUnsignedIntegerFeature(
    EFeature feature,
    Long low,
    Long high,
    EDoubleThresholdMode mode
)
```

Parameters

feature

The feature that serves as a filter.

threshold

The single threshold on the feature.

mode

Specifies the way the threshold is interpreted.

low

The low bound of the range on the feature.

high

The high bound of the range on the feature.

EObjectSelection.RenderMask

Creates a Flexible Mask from the selection.

[VB6]

```
void RenderMask(
    EROIBW8 result,
    Long offsetX,
    Long offsetY
)
```

```
void RenderMask(
    EROIBW8 result
)
```

Parameters*result*

The image in which the generated mask will be stored.

offsetX

The X-offset that must be applied to bring the zero X-coordinate in the coded image on the first column of the result image (defaults to zero).

offsetY

The Y-offset that must be applied to bring the zero Y-coordinate in the coded image on the first row of the result image (defaults to zero).

Remarks

The size of the result image will not be changed: It must be properly sized beforehand.

This method generates an exception if several layers are encoded, in which case no default layer exists.

EObjectSelection.Sort

Sorts the selected coded elements according to the value of a feature.

[VB6]

```
void Sort(
    EFeature feature
)

void Sort(
    EFeature feature,
    ESortDirection direction
)
```

Parameters*feature*

The feature to sort with.

direction

The sorting direction. By default, the features are sorted by increasing values.

EObjectSelection.UnsignedIntegerFeatureMaximum

Computes the maximum value of the given feature across the selection.

[VB6]

```
Long UnsignedIntegerFeatureMaximum(  
    EFeature feature  
)
```

Parameters

feature

The feature of interest.

EObjectSelection.UnsignedIntegerFeatureMinimum

Computes the minimum value of the given feature across the selection.

[VB6]

```
Long UnsignedIntegerFeatureMinimum(  
    EFeature feature  
)
```

Parameters

feature

The feature of interest.

EOCR Class

Manages a complete context for the font-dependent printed character reader implemented in EasyOCR.

PROPERTIES

AverageFirstCharDistance	-
CharSpacing	Spacing that separates characters.
CompareAspectRatio	Flag indicating whether the character aspect ratio is used to compute the recognition score.
CutLargeChars	Flag indicating whether the large chars cutting mode is used.
MatchingMode	Matching mode to use to compare characters to the template.
MaxCharHeight	Maximum character height.
MaxCharWidth	Maximum character width.
MinCharHeight	Minimum character height.

MinCharWidth	Minimum character width.
NoiseArea	Noise area.
NumChars	Number of recognized characters.
NumPatterns	Number of patterns in the current font.
PatternHeight	Current pattern height, as set at the last EOCR::NewFont or EOCR::Load operation.
PatternWidth	Current pattern width, as set at the last EOCR::NewFont or EOCR::Load operation.
RelativeSpacing	Relative spacing value.
RelativeThreshold	Relative threshold used for image segmentation.
RemoveBorder	Flag indicating whether all blobs touching the ROI edges are automatically discarded.
RemoveNarrowOrFlat	Flag indicating whether the characters are discarded when either dimension falls below the minimum value
SegmentationMode	Segmentation mode.

ShiftingMode	Shifting mode to use to compare characters to the template.
ShiftXTolerance	Horizontal translation tolerance around the nominal position when the character positions are explicitly specified.
ShiftYTolerance	Vertical translation tolerance around the nominal position when the character positions are explicitly specified.
TextColor	Text color.
Threshold	Threshold mode used for image segmentation.
TrueThreshold	M Absolute threshold level when using a single threshold. E
THODS	
AddChar	Add a character coordinates to the list.
AddPatternFromImage	Adds a new pattern to the font.
BuildObjects	Segments the source image, i.e. detects and labels the objects.

CharGetDstX	Returns the abscissa of the lower right corner of a recognized character.
CharGetDstY	Returns the ordinate of the lower right corner of a recognized character.
CharGetHeight	Returns the height of a recognized character.
CharGetOrgX	Returns the abscissa of the upper left corner of a recognized character.
CharGetOrgY	Returns the ordinate of the upper left corner of a recognized character.
CharGetTotalDstX	Returns the abscissa of the lower right corner of a recognized character.
CharGetTotalDstY	Returns the ordinate of the lower right corner of a recognized character.
CharGetTotalOrgX	Returns the abscissa of the upper left corner of a recognized character.
CharGetTotalOrgY	Returns the ordinate of the upper left corner of a recognized character.
CharGetWidth	Returns the width of a recognized character.

DrawChar	Draws the bounding box of a given character.
DrawChars	Draws the bounding boxes of all characters in the image.
DrawCharsWithCurrentPen	Draws the bounding boxes of all characters in the image.
DrawCharWithCurrentPen	Draws the bounding box of a given character.
EmptyChars	Empties the list of known characters.
EOCR	Constructs an OCR context.
FindAllChars	Locates the characters by filtering the objects according to their size, and grouping them if the segmentation mode is set to ESegmentationMode_RepasteObjects .
GetConfidenceRatio	Returns the degree of confidence in the recognized character.
GetFirstCharCode	Returns the code of the pattern that matches at best a recognized character.

[GetFirstCharDistance](#)

Computes the degree of similarity between the best matching pattern and a recognized character.

[GetPatternBitmap](#)

Returns a pointer to an image holding the pattern of the given index. This image should not be modified.

[GetPatternClass](#)

Returns the class of a given pattern in the current font.

[GetPatternCode](#)

Returns the character code of a given pattern in the current font.

[GetSecondCharCode](#)

Returns the code of the pattern that matches at second best a recognized character.

[GetSecondCharDistance](#)

Computes the degree of similarity between the second best matching pattern and a recognized character.

[HitChar](#)

Returns **TRUE** if the cursor is placed over the character specified by **charIndex**.

[HitChars](#)

Returns **TRUE** if the cursor is placed over a character and sets the **charIndex** accordingly.

[LearnPattern](#)

Adds a new pattern to the font.

LearnPatterns	Adds all the patterns from the source image to the font.
Load	-
MatchChar	Reads a single character, i.e. finds the best match between the patterns in the font and the given character.
NewFont	Empties the contents of the font and sets the size of the pattern array to be used from then on.
operator=	Copies all the data from another EOCR object into the current EOCR object
ReadText	Reads one or more rows of characters, i.e. finds the best match between the patterns in the font and the segmented characters.
ReadTextWide	Reads one or more rows of characters, i.e. finds the best match between the patterns in the font and the segmented characters.
Recognize	Achieves all processing phases (blob analysis, character segmentation and pattern recognition) in a single operation.

[RecognizeWide](#)

Achieves all processing phases (blob analysis, character segmentation and pattern recognition) in a single operation.

[RemovePattern](#)

Removes a given pattern from the current font.

[Save](#)

-

[SetPatternClass](#)

-

[SetPatternCode](#)

E-

O

CR.AddChar

Add a character coordinates to the list.

[VB6]

```
void AddChar(  
    Long originX,  
    Long originY,  
    Long endX,  
    Long endY  
)
```

Parameters

originX

Abscissa of the upper left corner of the bounding box of the character.

originY

Ordinate of the upper left corner of the bounding box of the character.

endX

Abscissa of the bottom right corner of the bounding box of the character.

endY

Ordinate of the bottom right corner of the bounding box of the character.

Remarks

It is to use when you know the exact position of the characters to be recognized, to bypass the segmentation step, for efficiency or reliability purposes. To use this feature, simply specify the character positions by successive calls to **AddChar**. When this is done, the remainder of the OCR processing steps can take place. To empty the list of known characters, call [EOCR::EmptyChars](#).

EOCR.AddPatternFromImage

Adds a new pattern to the font.

[VB6]

```
void AddPatternFromImage(
    EROIBW8 sourceImage,
    Long originX,
    Long originY,
    Long width,
    Long height,
    Long code,
    Long classes
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

originX

Abscissa of the upper left corner of the bounding box of the pattern.

originY

Ordinate of the upper left corner of the bounding box of the pattern.

width

Width of the bounding box of the pattern.

height

Height of the bounding box of the pattern.

code

Character code of the pattern.

classes

Class of the pattern, as defined by [EOCRClass](#).

Remarks

The pattern is extracted from an image by specifying a bounding rectangle.

EOCR.AverageFirstCharDistance

-

[VB6]

AverageFirstCharDistance As Single

read-only

EOCR.BuildObjects

Segments the source image, i.e. detects and labels the objects.

[VB6]

```
void BuildObjects(  
    EROIBW8 sourceImage  
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

Remarks

An object is a connected set of pixels above or below **Threshold** (according to **TextColor**).

EOCR.CharGetDstX

Returns the abscissa of the lower right corner of a recognized character.

[VB6]

```
Long CharGetDstX(  
    Long index  
)
```

Parameters

index

Character number (in range **0..NumChars-1**).

EOCR.CharGetDstY

Returns the ordinate of the lower right corner of a recognized character.

[VB6]

```
Long CharGetDstY(  
    Long index  
)
```

Parameters

index

Character number (in range **0..NumChars-1**).

EOCR.CharGetHeight

Returns the height of a recognized character.

[VB6]

```
Long CharGetHeight(  
    Long index  
)
```

Parameters

index

Character number (in range **0..NumChars-1**).

EOCR.CharGetOrgX

Returns the abscissa of the upper left corner of a recognized character.

[VB6]

```
Long CharGetOrgX(  
    Long index  
)
```

Parameters

index

Character number (in range **0..NumChars-1**).

EOCR.CharGetOrgY

Returns the ordinate of the upper left corner of a recognized character.

[VB6]

```
Long CharGetOrgY(  
    Long index  
)
```

Parameters

index

Character number (in range **0..NumChars-1**).

EOCR.CharGetTotalDstX

Returns the abscissa of the lower right corner of a recognized character.

[VB6]

```
Long CharGetTotalDstX(  
    Long index  
)
```

Parameters

index

Character number (in range **0..NumChars-1**).

Remarks

The coordinates are computed with respect to the parent image rather than the ROI.

EOCR.CharGetTotalDstY

Returns the ordinate of the lower right corner of a recognized character.

[VB6]

```
Long CharGetTotalDstY(  
    Long index  
)
```

Parameters

index

Character number (in range **0..NumChars-1**).

Remarks

The coordinates are computed with respect to the parent image rather than the ROI.

EOCR.CharGetTotalOrgX

Returns the abscissa of the upper left corner of a recognized character.

[VB6]

```
Long CharGetTotalOrgX(  
    Long index  
)
```

Parameters

index

Character number (in range **0..NumChars-1**).

Remarks

The coordinates are computed with respect to the parent image rather than the ROI.

EOCR.CharGetTotalOrgY

Returns the ordinate of the upper left corner of a recognized character.

[VB6]

```
Long CharGetTotalOrgY(  
    Long index  
)
```

Parameters

index

Character number (in range **0..NumChars-1**).

Remarks

The coordinates are computed with respect to the parent image rather than the ROI.

EOCR.CharGetWidth

Returns the width of a recognized character.

[VB6]

```
Long CharGetWidth(  
    Long index  
)
```

Parameters

index

Character number (in range **0..NumChars-1**).

EOCR.CharSpacing

Spacing that separates characters.

[VB6]

CharSpacing As Long

read-write

Remarks

When two objects are separated by a vertical gap larger or equal than the spacing, they are considered to belong to distinct characters. The segmentation parameters *must* be the same for both learning and recognition process.

EOCR.CompareAspectRatio

Flag indicating whether the character aspect ratio is used to compute the recognition score.

[VB6]

CompareAspectRatio As Boolean

read-write

Remarks

When **TRUE**, the character aspect ratio is used to compute the recognition score.

EOCR.CutLargeChars

Flag indicating whether the large chars cutting mode is used.

[VB6]

CutLargeChars As Boolean

read-write

Remarks

If **TRUE**, candidate characters larger than **MaxWidth** are split into as many parts as required.
If **FALSE**, large characters are discarded. The segmentation parameters *must* be the same for both learning and recognition process.

EOCR.DrawChar

Draws the bounding box of a given character.

[VB6]

```
void DrawChar(
    Long graphicContext,
    Long charIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawChar(
    Long graphicContext,
    ERGBColor color,
    Long charIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawChar(
    EDrawAdapter graphicContext,
    Long charIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

charIndex

Character index, in range **0..NumChars-1**.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used.

EOCR.DrawChars

Draws the bounding boxes of all characters in the image.

[VB6]

```
void DrawChars(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawChars(
    Long graphicContext,
    ERGBColor color,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawChars(
    EDrawAdapter graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used for all characters (to vary the colors, draw the objects separately using the [EOCR::DrawChar](#) method instead).

EOCR.DrawCharsWithCurrentPen

Draws the bounding boxes of all characters in the image.

[VB6]

```
void DrawCharsWithCurrentPen(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used for all characters (to vary the colors, draw the objects separately using the [EOCR::DrawChar](#) method instead).

EOCR.DrawCharWithCurrentPen

Draws the bounding box of a given character.

[VB6]

```
void DrawCharWithCurrentPen(
    Long graphicContext,
    Long charIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

charIndex

Character index, in range **0..NumChars-1**.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used.

EOCR.EmptyChars

Empties the list of known characters.

```
[VB6]
void EmptyChars(
)
```

Remarks

It is to use when you know the exact position of the characters to be recognized. See member [EOCR::AddChar](#).

EOCR.EOCR

Constructs an OCR context.

```
[VB6]
void EOCR(
)
void EOCR(
    EOCR other
)
```

Parameters

other

Another EOCR object to be copied in the new EOCR object.

Remarks

By default, the **Threshold** property is set to **128**.

EOCR.FindAllChars

Locates the characters by filtering the objects according to their size, and grouping them if the segmentation mode is set to [ESegmentationMode_RepasteObjects](#).

[VB6]

```
void FindAllChars(  
    EROIBW8 sourceImage  
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

Remarks

The characters are sorted from left to right. This operation must be performed after a call to [EOCR::BuildObjects](#) and before a call to [EOCR::ReadText](#).

EOCR.GetConfidenceRatio

Returns the degree of confidence in the recognized character.

[VB6]

```
Single GetConfidenceRatio(  
    Long index  
)
```

Parameters

index

Character number (in range **0..NumChars-1**).

Remarks

A value of 100 % means there is no difference between the recognized character and the best matching pattern. A value of 0 % means there is no way to distinguish between the best and second best matching pattern. The computation of the confidence ratio is based on the first and second characters distance parameters (see [EOCR::GetFirstCharDistance](#) and [EOCR::GetSecondCharDistance](#)).

EOCR.GetFirstCharCode

Returns the code of the pattern that matches at best a recognized character.

[VB6]

```
Long GetFirstCharCode(  
    Long index  
)
```

Parameters

index

Character number (in range **0..NumChars-1**).

EOCR.GetFirstCharDistance

Computes the degree of similarity between the best matching pattern and a recognized character.

[VB6]

```
Single GetFirstCharDistance(  
    Long index  
)
```

Parameters

index

Character number (in range **0..NumChars-1**).

Remarks

Returns **0** for a perfect match and **1** for a total mismatch.

EOCR.GetPatternBitmap

Returns a pointer to an image holding the pattern of the given index. This image should not be modified.

[VB6]

```
EImageBW8 GetPatternBitmap(  
    Long index  
)
```

Parameters

index

Pattern number (in range **0.. -1**).

EOCR.GetPatternClass

Returns the class of a given pattern in the current font.

[VB6]

```
Long GetPatternClass(  
    Long index  
)
```

Parameters

index

Pattern number (in range **0..NumPatterns-1**).

EOCR.GetPatternCode

Returns the character code of a given pattern in the current font.

[VB6]

```
Long GetPatternCode(  
    Long index  
)
```

Parameters

index

Pattern number (in range **0..NumPatterns-1**).

EOCR.GetSecondCharCode

Returns the code of the pattern that matches at second best a recognized character.

[VB6]

```
Long GetSecondCharCode(  
    Long index  
)
```

Parameters

index

Character number (in range **0..NumChars-1**).

EOCR.GetSecondCharDistance

Computes the degree of similarity between the second best matching pattern and a recognized character.

[VB6]

```
Single GetSecondCharDistance(
    Long index
)
```

Parameters*index*Character number (in range **0..NumChars-1**).**Remarks**Returns **0** for a perfect match and **1** for a total mismatch.

EOCR.HitChar

Returns **TRUE** if the cursor is placed over the character specified by **charIndex**.

[VB6]

```
Boolean HitChar(
    Long cursorX,
    Long cursorY,
    Long charIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters*cursorX*

Current cursor coordinates.

cursorY

Current cursor coordinates.

charIndex

Index of the character to be hit.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0**, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

Remarks

If zooming and/or panning were used when drawing the ROI, the same values must be used with [HitChar](#).

EOCR.HitChars

Returns **TRUE** if the cursor is placed over a character and sets the **charIndex** accordingly.

[VB6]

```
Boolean HitChars(
    Long cursorX,
    Long cursorY,
    Long charIndex,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

cursorX

Current cursor coordinates.

cursorY

Current cursor coordinates.

charIndex

Index of the character hit.

zoomX

Horizontal zooming factor. By default, **TRUE** scale is used.

zoomY

Vertical zooming factor. If set to **0**, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

Remarks

If zooming and/or panning were used when drawing the ROI, the same values must be used with [HitChars](#).

EOCR.LearnPattern

Adds a new pattern to the font.

[VB6]

```
void LearnPattern(
    EROIBW8 sourceImage,
    Long charIndex,
    Long code,
    Long classes,
    Boolean autoSegmentation
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

charIndex

Index of the character (ASCII or Unicode) to learn.

code

Character code of the pattern.

classes

Class of the pattern, as defined by [EOCRClass](#).

autoSegmentation

Boolean indicating whether the calculation of the true threshold has to be forced (default **TRUE**) or bypassed (**FALSE**).

Remarks

The pattern is selected by its index value, as assigned by the [EOCR::FindAllChars](#) process.

EOCR.LearnPatterns

Adds all the patterns from the source image to the font.

[VB6]

```
void LearnPatterns(
    EROIBW8 sourceImage,
    String text,
    Long singleClass,
    Boolean autoSegmentation
)

void LearnPatterns(
    EROIBW8 sourceImage,
    String text,
    ()Long classes,
    Boolean autoSegmentation
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

text

String containing character codes of the patterns.

singleClass

If specified, all the characters of the string are associated with the same class(es), that is specified by **singleClass**.

autoSegmentation

Boolean indicating whether the calculation of the true threshold has to be forced (default **TRUE**) or bypassed (**FALSE**).

classes

If specified, the i-th character of the string is associated with the class specified by the i-th element of the vector **classes**.

Remarks

Patterns are ordered by their index value, as assigned by the [EOCR::FindAllChars](#) process.

EOCR.Load

-

[VB6]

```
void Load(  
    ESerializer serializer  
)  
  
void Load(  
    String stream  
)
```

Parameters

serializer

-

stream

EOCR.MatchChar

Reads a single character, i.e. finds the best match between the patterns in the font and the given character.

[VB6]

```
void MatchChar(  
    EROIBW8 sourceImage,  
    Long classes,  
    Long index,  
    Long shiftX,  
    Long shiftY  
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

classes

Logical mask obtained by combining the values of [EOCRClass](#), to specify to what classes the character may belong.

index

Character number (in range **0..NumChars-1**).

shiftX

Horizontal translation applied to the character.

shiftY

Vertical translation applied to the character.

Remarks

A shift can be apply to the character. This operation can only be performed after a call to [EOCR::FindAllChars](#).

EOCR.MatchingMode

Matching mode to use to compare characters to the template.

[VB6]

MatchingMode As EMatchingMode

read-write

Remarks

By default, the root-mean-square error method is used. These modes are meant to be used without thresholding, that is when one of the [EOCRColor_DarkOnLight](#) and [EOCRColor_LightOnDark](#) colors are used.

EOCR.MaxCharHeight

Maximum character height.

[VB6]

MaxCharHeight As Long

read-write

Remarks

A character larger than the maximum width or higher than the maximum height is discarded.
The segmentation parameters *must* be the same for both learning and recognition process.

EOCR.MaxCharWidth

Maximum character width.

[VB6]

MaxCharWidth As Long

read-write

Remarks

A character larger than the maximum width or higher than the maximum height is discarded.
The segmentation parameters *must* be the same for both learning and recognition process.

EOCR.MinCharHeight

Minimum character height.

[VB6]

MinCharHeight As Long

read-write

Remarks

A character both narrower than the minimum width and lower than the minimum height is discarded. The segmentation parameters *must* be the same for both learning and recognition process.

EOCR.MinCharWidth

Minimum character width.

[VB6]

MinCharWidth As Long

read-write

Remarks

A character both narrower than the minimum width and lower than the minimum height is discarded. The segmentation parameters *must* be the same for both learning and recognition process.

EOCR.NewFont

Empties the contents of the font and sets the size of the pattern array to be used from then on.

[VB6]

```
void NewFont(
    Long patternWidth,
    Long patternHeight
)
```

Parameters

patternWidth

Width of the normalized pattern representation, in pixels.

patternHeight

Height of the normalized pattern representation, in pixels.

Remarks

A larger pattern array improves the recognition sensitivity, at the expense of increased processing time.

EOCR.NoiseArea

Noise area.

[VB6]

NoiseArea As Long

read-write

Remarks

When a blob has an area smaller than this value, it is ignored. The segmentation parameters *must* be the same for both learning and recognition process.

EOCR.NumChars

Number of recognized characters.

[VB6]

NumChars As Long

read-only

EOCR.NumPatterns

Number of patterns in the current font.

[VB6]

NumPatterns As Long

read-only

EOCR.operator=

Copies all the data from another EOCR object into the current EOCR object

[VB6]

```
EOCR operator=(  
    EOCR other  
)
```

Parameters

other

EOCR object to be copied

EOCR.PatternHeight

Current pattern height, as set at the last [EOCR::NewFont](#) or [EOCR::Load](#) operation.

[VB6]

PatternHeight As Long

read-only

EOCR.PatternWidth

Current pattern width, as set at the last [EOCR::NewFont](#) or [EOCR::Load](#) operation.

[VB6]

PatternWidth As Long

read-only

EOCR.ReadText

Reads one or more rows of characters, i.e. finds the best match between the patterns in the font and the segmented characters.

[VB6]

```
String ReadText(
    EROIBW8 sourceImage,
    Long maximumNumberOfCharacters,
    Long classes,
    Boolean autoSegmentation
)

String ReadText(
    EROIBW8 sourceImage,
    Long maximumNumberOfCharacters,
    () Long classes,
    Boolean autoSegmentation
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

maximumNumberOfCharacters

Maximum number of characters to be read.

classes

Pointer to an array of logical masks obtained by combining the values of [EOCRClass](#), to specify to what classes the character may belong. Each mask value in the array applies to the corresponding character.

autoSegmentation

Boolean indicating whether the calculation of the true threshold has to be forced (default **TRUE**) or bypassed (**FALSE**).

Remarks

This operation can only be performed after a call to [EOCR::FindAllChars](#). In case the text contains character with a code > 255, [ReadText](#) will throw an exception. In this case, you should use [EOCR::ReadTextWide](#).

EOCR.ReadTextWide

Reads one or more rows of characters, i.e. finds the best match between the patterns in the font and the segmented characters.

[VB6]

```
String ReadTextWide(
    EROIBW8 sourceImage,
    Long maximumNumberOfCharacters,
    Long classes,
    Boolean autoSegmentation
)

String ReadTextWide(
    EROIBW8 sourceImage,
    Long maximumNumberOfCharacters,
    () Long classes,
    Boolean autoSegmentation
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

maximumNumberOfCharacters

Maximum number of characters to be read.

classes

Pointer to an array of logical masks obtained by combining the values of [EOCRClass](#), to specify to what classes the character may belong. Each mask value in the array applies to the corresponding character.

autoSegmentation

Boolean indicating whether the calculation of the true threshold has to be forced (default **TRUE**) or bypassed (**FALSE**).

Remarks

This operation can only be performed after a call to [EOCR::FindAllChars](#). In case the text contains character with a code > 65535, ReadTextWide will throw an exception. In this case, you should read the text character by character by using [EOCR::GetFirstCharCode](#).

EOCR.Recognize

Achieves all processing phases (blob analysis, character segmentation and pattern recognition) in a single operation.

[VB6]

```
String Recognize(
    EROIBW8 sourceImage,
    Long maximumNumberOfCharacters,
    Long classes
)

String Recognize(
    EROIBW8 sourceImage,
    Long maximumNumberOfCharacters,
    () Long classes
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

maximumNumberOfCharacters

Maximum number of characters to be read.

classes

Pointer to an array of logical mask obtained by combining the values of [EOCRClass](#), to specify to what classes the character may belong. Each mask value in the array applies to the corresponding character.

Remarks

This method does the same as a sequence of [EOCR::BuildObjects](#)/[EOCR::FindAllChars](#)/[EOCR::ReadText](#),

EOCR.RecognizeWide

Achieves all processing phases (blob analysis, character segmentation and pattern recognition) in a single operation.

[VB6]

```
String RecognizeWide(
    EROIBW8 sourceImage,
    Long maximumNumberOfCharacters,
    Long classes
)

String RecognizeWide(
    EROIBW8 sourceImage,
    Long maximumNumberOfCharacters,
    () Long classes
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

maximumNumberOfCharacters

Maximum number of characters to be read.

classes

Pointer to an array of logical mask obtained by combining the values of [EOCRClass](#), to specify to what classes the character may belong. Each mask value in the array applies to the corresponding character.

Remarks

This method does the same as a sequence of [EOCR::BuildObjects](#)/[EOCR::FindAllChars](#)/[EOCR::ReadText](#),

EOCR.RelativeSpacing

Relative spacing value.

[VB6]

RelativeSpacing As Single

read-write

Remarks

This value is the ratio of the width of the spaces between characters and the character width. For example, characters 25 pixels wide separated by 10 pixels gaps correspond to a relative spacing of **10/25 = 0.4**. The default value of this parameter is **0**, corresponding to no spacing. This parameter is relevant only when the **CutLargeChars** mode is enabled. The segmentation parameters *must* be the same for both learning and recognition process.

EOCR.RelativeThreshold

Relative threshold used for image segmentation.

[VB6]

RelativeThreshold As Single

read-write

Remarks

The **RelativeThreshold** is the fraction of the image pixels that will be set below the threshold. Only used when the [EOCR::Threshold](#) property is set to EThresholdMode_Relative. The default value is 0.5.

EOCR.RemoveBorder

Flag indicating whether all blobs touching the ROI edges are automatically discarded.

[VB6]

RemoveBorder As Boolean

read-write

Remarks

If **TRUE**, all blobs touching the ROI edges are automatically discarded. By default, this feature is turned on. The segmentation parameters *must* be the same for both learning and recognition process.

EOCR.RemoveNarrowOrFlat

Flag indicating whether the characters are discarded when either dimension falls below the minimum value

[VB6]

RemoveNarrowOrFlat As Boolean

read-write

Remarks

If **TRUE**, characters are discarded if either their width or their height is smaller than the minimum value. By default, this feature is disabled (only smaller characters in *both* height and width are discarded – the condition could be written **Narrow**<bc>**And**</bc>**Flat**). The segmentation parameters *must* be the same for both learning and recognition process.

EOCR.RemovePattern

Removes a given pattern from the current font.

[VB6]

```
void RemovePattern(  
    Long index  
)
```

Parameters

index

Index of the pattern to be removed from the current font.

EOCR.Save

-

```
[VB6]  
  
void Save(  
    ESerializer serializer  
)  
  
void Save(  
    String stream  
)
```

Parameters

serializer

-

stream

EOCR.SegmentationMode

Segmentation mode.

[VB6]

SegmentationMode As ESegmentationMode

read-write

Remarks

The segmentation parameters *must* be the same for both learning and recognition process.

EOCR.SetPatternClass

-

[VB6]

```
void SetPatternClass(  
    Long n32Index,  
    Long un32Class  
)
```

Parameters

n32Index

un32Class

-

EOCR.SetPatternCode

-

[VB6]

```
void SetPatternCode(  
    Long n32Index,  
    Long n32Code  
)
```

Parameters

n32Index

-

n32Code

-

EOCR.ShiftingMode

Shifting mode to use to compare characters to the template.

[VB6]

ShiftingMode As EShiftingMode

read-write

EOCR.ShiftXTolerance

Horizontal translation tolerance around the nominal position when the character positions are explicitly specified.

[VB6]

ShiftXTolerance As Long

read-write

Remarks

By default, no shifting is allowed (**ShiftX = 0**). The tolerance can be used in two ways: either each character is moved individually until the best match is found, or the set of characters is matched as a whole, until the best average error is reached. See [EOCR::ShiftingMode](#).

EOCR.ShiftYTolerance

Vertical translation tolerance around the nominal position when the character positions are explicitly specified.

[VB6]

ShiftYTolerance As Long

read-write

Remarks

By default, no shifting is allowed (**ShiftY = 0**). The tolerance can be used in two ways: either each character is moved individually until the best match is found, or the set of characters is matched as a whole, until the best average error is reached. See [EOCR::ShiftingMode](#).

EOCR.TextColor

Text color.

[VB6]

TextColor As EOCRColor

read-write

Remarks

The segmentation parameters *must* be the same for both learning and recognition process.

EOCR.Threshold

Threshold mode used for image segmentation.

[VB6]

Threshold As Long

read-write

Remarks

Threshold value as defined by the EThresholdMode enum. By default, the "minimum residue" mode is used to determine the threshold value. In case of an absolute threshold, the threshold value is given instead.

EOCR.TrueThreshold

Absolute threshold level when using a single threshold.

[VB6]

TrueThreshold As Long

read-only

Remarks

This value is valid only when the [EOCR::BuildObjects](#) or [EOCR::ReadText](#) method has been called beforehand.

EOCR2 Class

Manages a complete context for the font-dependent printed character reader implemented in EasyOCR2.

PROPERTIES

CharsHeight

Sets the expected character height in pixels.

CharsMaxFragmentation

Sets the **CharsMaxFragmentation** parameter for the segmentation algorithm. This will determine the minimum size a blob should be in order to be considered a potential character. A high setting will allow only larger blobs, a low setting will also allow smaller blobs. This parameter should be set between 0.0 and 1.0, the default setting is 0.1. The minimum blob size to be considered a potential character is defined as $\text{CharMaxFrag} * \text{CharHeight} * \text{CharWidth}$.

CharsSpacingBias

Sets the **CharSpacingBias** parameter for the topology fitting algorithm, which optimises the spacing of the bounding boxes to optimally fit the detected blobs. This will determine whether the method is biased toward finding narrow spacing, wide spacing or is neutral. The default setting for this parameter is [EasyOCR2CharSpacingBias_Neutral](#)

CharsWidth

Sets the expected character width in pixels.

[CharsWidthBias](#)

Sets the **CharsWidthBias** parameter for the topology fitting algorithm, which optimises the width of the bounding boxes to optimally fit the detected blobs. This will determine whether the method is biased toward finding narrow boxes, wide boxes or is neutral. The default setting for this parameter is [EasyOCR2CharWidthBias_Neutral](#)

[CharsWidthTolerance](#)

Sets the **CharsWidthTolerance** parameter for the topology fitting algorithm, which will attempt to optimise the width of the bounding boxes to optimally fit the detected blobs. This will determine the range of bounding box sizes that will be accepted, defined as: $(1 - \text{CharWidthTolerance}) * \text{CharWidth} \leq \text{BBoxWidth} \leq (1 + \text{CharWidthTolerance}) * \text{CharWidth}$. The CharWidthTolerance parameter should be between 0.0 and 1.0, the default setting is 0.25.

[DetectionDelta](#)

Sets the **DetectionDelta** parameter for the segmentation algorithm. This will determine the the range of grayscale-values used to determine the stability of a blob. A low setting will make the algorithm more sensitive to noise, a high setting will make the algorithm insensitive to blobs with low contrast to the background. This parameter should be set between 0 and 127, the default setting is 12.

[MaxVariation](#)

Sets the **maxVariation** parameter for the segmentation algorithm. This parameter determines how stable a blob in the image should be in order to be considered a potential character, a region with clearly defined edges is generally considered stable while a blurry region is not. A high setting allows more unstable blobs, a low setting allows only very stable blobs. This parameter should be set between 0.0 and 1.0, the default setting is 0.25.

[ReadText](#)

Outputs an [EOCR2Text](#) structure containing the detailed detection and recognition results.

[TextAngleTolerance](#)

Sets the **TextAngleTolerance** parameter for the topology fitting algorithm, which finds the angle of the text in the image with respect to the horizontal. This will limit the range of angles that will be tested, defined as: BaseAngle - AngleTolerance <= angle <= BaseAngle + AngleTolerance. This function follows the angle unit convention defined by [Easy::AngleUnit](#). The default setting for this parameter is 20°.

[TextBaseAngle](#)

Sets the **TextBaseAngle** parameter for the topology fitting algorithm, which will attempt to find the angle of the text in the image with respect to the horizontal. This will determine the center of the range of angles that will be tested, defined as: BaseAngle - AngleTolerance <= angle <= BaseAngle + AngleTolerance. This function follows the angle unit convention defined by [Easy::AngleUnit](#). The default setting for this parameter is 0°.

[TextPolarity](#)

Sets the **TextPolarity** parameter for the segmentation algorithm. This will determine whether the algorithm searches for light blobs in a dark background or for dark blobs in a light background. Default setting is `EasyOCR2TextPolarity_WhiteOnBlack`.

[Topology](#)**METHODS**

Sets the topology of the text that should be found in the image. A modified version of Regex expressions are used, where: .(dot) represents any character (not including a space). L represents an alphabetic character. Lu represents an uppercase letter. Ll represents a lowercase letter. N represents a digit. P represents a punctuation mark such as a bracket, quote, hyphen, dash, slash etc. S represents a math symbol or currency sign. \n represents a line break. space represents a space between two words. Combinations can be made, for example: [LN] represents an alpha-numeric character. To specify multiple characters, simply add {n} at the end for n characters. If the amount of characters is uncertain, specify {n,m} for a minimum of n characters and a maximum of m characters. The topology "[LuN]{3,5}PN{4} \n .{5} LL" represents a text comprised of 2 lines: - The first line has 1 word composed of 3 to 5 capital alpha-numeric characters, followed by a punctuation character and 4 digits. - The second line has 2 words. The first word comprises of 5 wildcard characters, the second word has 2 alphabetic characters (capital or small).

[AddCharactersToDatabase](#)

Reads reference characters from disk and adds them to the database used for text recognition. The characters can be read from a trueType (.ttf/.ttc) file or from an EasyOCR2 database file.

ClearCharacterDatabase	Clears the reference character database from this EOCR2 instance.
Detect	Finds the text in an image: - detects potential characters in the image following the given text polarity. - fits bounding boxes to the detected characters, following the given topology and character width/height. - extracts the detected characters from the image. The detected characters are output as an EOCR2text structure.
DrawDetection	Draws the bounding boxes found by the topology detection algorithm.
DrawDetectionWithCurrentPen	-
DrawRecognition	Draws the recognized text next to the character bounding box in the image.
DrawRecognitionWithCurrentPen	-
DrawSegmentation	Draws the blobs found by the segmentation algorithm.
DrawSegmentationWithCurrentPen	-
EOCR2	Constructs an EOCR2 context.

HitTestChar

Tests the cursor position for the presence of a character. If one is present under the cursor, it returns true and fills the EOCRChar object passed as parameter.

HitTestLine

Tests the cursor position for the presence of a line. If one is present under the cursor, it returns true and fills the EOCRLine object passed as parameter.

HitTestText

Tests the cursor position for the presence of a text. If one is present under the cursor, it returns true and fills the EOCRTText object passed as parameter.

HitTestWord

Tests the cursor position for the presence of a word. If one is present under the cursor, it returns true and fills the EOCRWord object passed as parameter.

Learn

Learns reference characters from a given **EOCR2Text/EOCR2Line/EOCR2Word/EOCR2Char** instance, containing user-specified character codes. The EOCR2Text/Line/Word/Char should contain detected characters from a reference image as well as their corresponding character codes.

Load

Loads a model, containing parameter settings used for all operations in **EOCR2**, from disk.

[Read](#)

Performs all steps required for reading text from an image: - detects potential characters in the image following the given text polarity. - fits bounding boxes to the detected characters, following the given topology and character width/height. - finds the optimal match between the detected characters and a given reference font. The read text is output as a string.

[Recognize](#)

Recognizes the characters in a given EOCR2text instance, based on a given reference font.

[Save](#)

Saves the model to disk, containing the current parameter setting used for all operations in [EOCR2](#).

[SaveCharacterDatabase](#)

Saves the current reference character database to disk.

CR2.AddCharactersToDatabase

Reads reference characters from disk and adds them to the database used for text recognition. The characters can be read from a trueType (.ttf/.ttc) file or from an EasyOCR2 database file.

[VB6]

```

void AddCharactersToDatabase(
    String file
)

void AddCharactersToDatabase(
    String file,
    EasyOCR2CharacterFilter filter
)

```

Parameters*file*

A string containing the path of the file.

filter

Optional parameter; an [EasyOCR2CharacterFilter](#) that tells the method which subset of the character-set should be loaded.

EOCR2.ChrsHeight

Sets the expected character height in pixels.

[VB6]

CharsHeight As Long

read-write

EOCR2.ChrsMaxFragmentation

Sets the **CharsMaxFragmentation** parameter for the segmentation algorithm. This will determine the minimum size a blob should be in order to be considered a potential character. A high setting will allow only larger blobs, a low setting will also allow smaller blobs. This parameter should be set between 0.0 and 1.0, the default setting is 0.1. The minimum blob size to be considered a potential character is defined as CharMaxFrag * CharHeight * CharWidth.

[VB6]

CharsMaxFragmentation As Single

read-write

EOCR2.ChrsSpacingBias

Sets the **CharSpacingBias** parameter for the topology fitting algorithm, which optimises the spacing of the bounding boxes to optimally fit the detected blobs. This will determine whether the method is biased toward finding narrow spacing, wide spacing or is neutral. The default setting for this parameter is [EasyOCR2CharSpacingBias_Neutral](#)

[VB6]

CharsSpacingBias As EasyOCR2CharSpacingBias

read-write

EOCR2.ChrsWidth

Sets the expected character width in pixels.

[VB6]

CharsWidth As Long

read-write

EOCR2.ChrsWidthBias

Sets the **CharsWidthBias** parameter for the topology fitting algorithm, which optimises the width of the bounding boxes to optimally fit the detected blobs. This will determine whether the method is biased toward finding narrow boxes, wide boxes or is neutral. The default setting for this parameter is [EasyOCR2CharWidthBias_Neutral](#)

[VB6]

CharsWidthBias As EasyOCR2CharWidthBias

read-write

EOCR2.ChrsWidthTolerance

Sets the **CharsWidthTolerance** parameter for the topology fitting algorithm, which will attempt to optimise the width of the bounding boxes to optimally fit the detected blobs. This will determine the range of bounding box sizes that will be accepted, defined as: $(1 - \text{CharWidthTolerance}) * \text{CharWidth} \leq \text{BBoxWidth} \leq (1 + \text{CharWidthTolerance}) * \text{CharWidth}$. The CharWidthTolerance parameter should be between 0.0 and 1.0, the default setting is 0.25.

[VB6]

CharsWidthTolerance As Single

read-write

EOCR2.ClearCharacterDatabase

Clears the reference character database from this EOCR2 instance.

[VB6]

```
void ClearCharacterDatabase()
)
```

EOCR2.Detect

Finds the text in an image: - detects potential characters in the image following the given text polarity. - fits bounding boxes to the detected characters, following the given topology and character width/height. - extracts the detected characters from the image. The detected characters are output as an EOCR2text structure.

[VB6]

```
EOCR2Text Detect(
    EROIBW8 srcRoi
)
```

Parameters

srcRoi

Pointer to the source image/ROI.

Remarks

The variables Topology, CharHeight, CharWidth should be set before performing this operation.

EOCR2.DetectionDelta

Sets the **DetectionDelta** parameter for the segmentation algorithm. This will determine the the range of grayscale-values used to determine the stability of a blob. A low setting will make the algorithm more sensitive to noise, a high setting will make the algorithm insensitive to blobs with low contrast to the background. This parameter should be set between 0 and 127, the default setting is 12.

[VB6]

DetectionDelta As Long

read-write

EOCR2.DrawDetection

Draws the bounding boxes found by the topology detection algorithm.

[VB6]

```
void DrawDetection(
    Long hdc,
    EasyOCR2DrawDetectionStyle style,
    Single zoomX,
    Single zoomY,
    Long panX,
    Long panY
)

void DrawDetection(
    Long hdc,
    ERGBColor color,
    EasyOCR2DrawDetectionStyle style,
    Single zoomX,
    Single zoomY,
    Long panX,
    Long panY
)

void DrawDetection(
    EDrawAdapter drawAdapter,
    EasyOCR2DrawDetectionStyle style,
    Single zoomX,
    Single zoomY,
    Long panX,
    Long panY
)
```

Parameters

hdc

Handle of the device context on which to draw.

style

The style in which each detection box should be drawn.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

color

The color in which to draw the detection.

drawAdapter

-

EOCR2.DrawDetectionWithCurrentPen

-

[VB6]

```
void DrawDetectionWithCurrentPen(
    Long hdc,
    EasyOCR2DrawDetectionStyle style,
    Single zoomX,
    Single zoomY,
    Long panX,
    Long panY
)
```

Parameters

hdc

-

style

-

zoomX

-

zoomY

-

panX

-

panY

-

EOCR2.DrawRecognition

Draws the recognized text next to the character bounding box in the image.

[VB6]

```
void DrawRecognition(
    Long hdc,
    EasyOCR2DrawRecognitionStyle style,
    Long cHeight,
    Single zoomX,
    Single zoomY,
    Long panX,
    Long panY
)

void DrawRecognition(
    EDrawAdapter drawAdapter,
    EasyOCR2DrawRecognitionStyle style,
    Long cHeight,
    Single zoomX,
    Single zoomY,
    Long panX,
    Long panY
)
```

```

void DrawRecognition(
    Long hdc,
    ERGBColor textColor,
    ERGBColor backgroundColor,
    EasyOCR2DrawRecognitionStyle style,
    Long cHeight,
    Single zoomX,
    Single zoomY,
    Long panX,
    Long panY
)

void DrawRecognition(
    EDrawAdapter drawAdapter,
    ERGBColor textColor,
    ERGBColor backgroundColor,
    EasyOCR2DrawRecognitionStyle style,
    Long cHeight,
    Single zoomX,
    Single zoomY,
    Long panX,
    Long panY
)

```

Parameters

hdc

Handle of the device context on which to draw.

style

The style in which each recognition result should be drawn.

cHeight

The character-height with which the recognized text should be displayed.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

drawAdapter

textColor

The color in which to draw the recognized text.

backgroundColor

The color of the box in which the text is displayed.

EOCR2.DrawRecognitionWithCurrentPen

-

[VB6]

```
void DrawRecognitionWithCurrentPen(
    Long hdc,
    EasyOCR2DrawRecognitionStyle style,
    Long cHeight,
    Single zoomX,
    Single zoomY,
    Long panX,
    Long panY
)
```

Parameters

hdc

-

style

-

cHeight

-

zoomX

-

zoomY

-

panX

-

panY

-

EOCR2.DrawSegmentation

Draws the blobs found by the segmentation algorithm.

[VB6]

```
void DrawSegmentation(
    Long hdc,
    EasyOCR2DrawSegmentationStyle style,
    Single zoomX,
    Single zoomY,
    Long panX,
    Long panY
)

void DrawSegmentation(
    Long hdc,
    ERGBColor color,
    EasyOCR2DrawSegmentationStyle style,
    Single zoomX,
    Single zoomY,
    Long panX,
    Long panY
)

void DrawSegmentation(
    EDrawAdapter drawAdapter,
    EasyOCR2DrawSegmentationStyle style,
    Single zoomX,
    Single zoomY,
    Long panX,
    Long panY
)
```

Parameters

hdc

Handle of the device context on which to draw.

style

The style in which each blob should be drawn.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

color

The color in which to draw the segmentation.

drawAdapter

-

Remarks

- When style is set to EasyOCR2DrawSegmentationStyle_DrawBlobs, the function draws the detected blobs
- When style is set to EasyOCR2DrawSegmentationStyle_DrawEllipses, the function draws a representative ellipse for each blob

EOCR2.DrawSegmentationWithCurrentPen

-

[VB6]

```
void DrawSegmentationWithCurrentPen(
    Long hdc,
    EasyOCR2DrawSegmentationStyle style,
    Single zoomX,
    Single zoomY,
    Long panX,
    Long panY
)
```

Parameters

hdc

-

style

-

zoomX

-

zoomY

-
panX

-
panY

EOCR2.EOCR2

Constructs an EOCR2 context.

```
[VB6]  
void EOCR2(  
)
```

EOCR2.HitTestChar

Tests the cursor position for the presence of a character. If one is present under the cursor, it returns true and fills the EOCRChar object passed as parameter.

```
[VB6]  
Boolean HitTestChar(  
EOCR2Char character,  
Long lineN,  
Long wordN,  
Long charN,  
Long x,  
Long y,  
Single zoomX,  
Single zoomY,  
Long panX,  
Long panY  
)
```

Parameters

character

Returns the character if one was detected under the cursor.

lineN

Returns the line-number of the character if one was detected under the cursor.

wordN

Returns the word-number of the character if one was detected under the cursor.

charN

Returns the character-number of the character if one was detected under the cursor.

x

Horizontal position of the cursor.

y

Vertical position of the cursor.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

EOCR2.HitTestLine

Tests the cursor position for the presence of a line. If one is present under the cursor, it returns true and fills the EOCRLine object passed as parameter.

[VB6]

```
Boolean HitTestLine(
    EOCR2Line line,
    Long x,
    Long y,
    Single zoomX,
    Single zoomY,
    Long panX,
    Long panY
)
```

Parameters

line

Object to fill if a line was detected under the cursor.

x

Horizontal position of the cursor.

y

Vertical position of the cursor.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

EOCR2.HitTestText

Tests the cursor position for the presence of a text. If one is present under the cursor, it returns true and fills the EOCRText object passed as parameter.

[VB6]

```
Boolean HitTestText(  
    EOCR2Text text,  
    Long x,  
    Long y,  
    Single zoomX,  
    Single zoomY,  
    Long panX,  
    Long panY  
)
```

Parameters

text

Object to fill if a text was detected under the cursor.

x

Horizontal position of the cursor.

y

Vertical position of the cursor.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

EOCR2.HitTestWord

Tests the cursor position for the presence of a word. If one is present under the cursor, it returns true and fills the EOCRWord object passed as parameter.

[VB6]

```
Boolean HitTestWord(
    EOCR2Word word,
    Long x,
    Long y,
    Single zoomX,
    Single zoomY,
    Long panX,
    Long panY
)
```

Parameters

word

Object to fill if a word was detected under the cursor.

x

Horizontal position of the cursor.

y

Vertical position of the cursor.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

EOCR2.Learn

Learns reference characters from a given [EOCR2Text](#)/[EOCR2Line](#)/[EOCR2Word](#)/[EOCR2Char](#) instance, containing user-specified character codes. The EOCR2Text/Line/Word/Char should contain detected characters from a reference image as well as their corresponding character codes.

[VB6]

```

void Learn(
    EOCR2Char character
)

void Learn(
    EOCR2Word word
)

void Learn(
    EOCR2Line line
)

void Learn(
    EOCR2Text text
)

```

Parameters

character

A single [EOCR2Char](#) character, containing the detected character from the reference image as well as the corresponding character code.

word

A single [EOCR2Word](#) word, containing the detected characters in a single word from the reference image as well as the corresponding character codes.

line

A single [EOCR2Line](#) line, containing the detected characters in a single line from the reference image as well as the corresponding character codes.

text

A complete [EOCR2Text](#) text, containing all detected characters from the reference image as well as the corresponding character codes.

EOCR2.Load

Loads a model, containing parameter settings used for all operations in [EOCR2](#), from disk.

[VB6]

```

void Load(
    String modelPath
)

```

Parameters*modelPath*

A string containing the full path to the model file.

EOCR2.MaxVariation

Sets the **maxVariation** parameter for the segmentation algorithm. This parameter determines how stable a blob in the image should be in order to be considered a potential character, a region with clearly defined edges is generally considered stable while a blurry region is not. A high setting allows more unstable blobs, a low setting allows only very stable blobs. This parameter should be set between 0.0 and 1.0, the default setting is 0.25.

[VB6]

MaxVariation As Single

read-write

EOCR2.Read

Performs all steps required for reading text from an image: - detects potential characters in the image following the given text polarity. - fits bounding boxes to the detected characters, following the given topology and character width/height. - finds the optimal match between the detected characters and a given reference font. The read text is output as a string.

[VB6]

```
String Read(
    EROIBW8 srcRoi
)
```

Parameters*srcRoi*

Pointer to the source image/ROI.

Remarks

The variables TextPolarity, Topology, CharHeight, CharWidth should be set and a reference font should be set before performing this operation.

EOCR2.ReadText

Outputs an [EOCR2Text](#) structure containing the detailed detection and recognition results.

[VB6]

ReadText As EOCR2Text

read-only

EOCR2.Recognize

Recognizes the characters in a given EOCR2text instance, based on a given reference font.

[VB6]

```
String Recognize(
    EOCR2Text text
)

String Recognize(
    EOCR2Text text,
    EROIBW8 srcRoi
)
```

Parameters

text

EOCR2text structure containing the detected characters from the image.

srcRoi

Pointer to the source image/ROI.

Remarks

A reference font should be provided before performing this operation. The overloaded function that uses an ROI is not yet implemented.

EOCR2.Save

Saves the model to disk, containing the current parameter setting used for all operations in EOCR2.

```
[VB6]  
void Save(  
    String modelPath  
)
```

Parameters

modelPath

A string containing the full path to the model file.

Remarks

It is advised to use a file extension that is non-standard (for instance *.ocr2)

EOCR2.SaveCharacterDatabase

Saves the current reference character database to disk.

```
[VB6]  
void SaveCharacterDatabase(  
    String file  
)
```

Parameters

file

A string containing the path of the file.

EOCR2.TextAngleTolerance

Sets the **TextAngleTolerance** parameter for the topology fitting algorithm, which finds the angle of the text in the image with respect to the horizontal. This will limit the range of angles that will be tested, defined as: $\text{BaseAngle} - \text{AngleTolerance} \leq \text{angle} \leq \text{BaseAngle} + \text{AngleTolerance}$. This function follows the angle unit convention defined by [Easy::AngleUnit](#). The default setting for this parameter is 20° .

[VB6]

```
TextAngleTolerance As Single
```

read-write

EOCR2.TextBaseAngle

Sets the **TextBaseAngle** parameter for the topology fitting algorithm, which will attempt to find the angle of the text in the image with respect to the horizontal. This will determine the center of the range of angles that will be tested, defined as: $\text{BaseAngle} - \text{AngleTolerance} \leq \text{angle} \leq \text{BaseAngle} + \text{AngleTolerance}$. This function follows the angle unit convention defined by [Easy::AngleUnit](#). The default setting for this parameter is 0° .

[VB6]

```
TextBaseAngle As Single
```

read-write

EOCR2.TextPolarity

Sets the **TextPolarity** parameter for the segmentation algorithm. This will determine whether the algorithm searches for light blobs in a dark background or for dark blobs in a light background. Default setting is [EasyOCR2TextPolarity_WhiteOnBlack](#).

[VB6]

TextPolarity As EasyOCR2TextPolarity

read-write

EOCR2.Topology

Sets the topology of the text that should be found in the image. A modified version of Regex expressions are used, where: .(dot) represents any character (not including a space). L represents an alphabetic character. Lu represents an uppercase letter. LL represents a lowercase letter. N represents a digit. P represents a punctuation mark such as a bracket, quote, hyphen, dash, slash etc. S represents a math symbol or currency sign. \n represents a line break. space represents a space between two words. Combinations can be made, for example: [LN] represents an alpha-numeric character. To specify multiple characters, simply add {n} at the end for n characters. If the amount of characters is uncertain, specify {n,m} for a minimum of n characters and a maximum of m characters. The topology "[LuN]{3,5}PN{4} \n .{5} LL" represents a text comprised of 2 lines: - The first line has 1 word composed of 3 to 5 capital alpha-numeric characters, followed by a punctuation character and 4 digits. - The second line has 2 words. The first word comprises of 5 wildcard characters, the second word has 2 alphabetic characters (capital or small).

[VB6]

Topology As String

read-write

EOCR2Char Class

Holds all information related to a single detected character.

PROPERTIES

Bitmap

Returns the bitmap associated to this character.

BoundingBox	Returns the bounding box associated to this character.
Candidates	Retrieves the list of recognition results for all reference-characters with their respective scores.
Text	Sets the value of the character, this is used during the learning phase.
TextCode	M Sets the value of the character, this is used during the learning phase. E
THODS	
EOCR2Char	Constructs an EOCR2Char context.
operator=	EOCR2Char.Bitmap
	<small>Creates a new context from another EOCR2Char object.</small>
Returns the bitmap associated to this character.	

[VB6]

Bitmap As EROIBW8

read-only

EOCR2Char.BoundingBox

Returns the bounding box associated to this character.

[VB6]

BoundingBox As ERectangle

read-only

EOCR2Char.Candidates

Retrieves the list of recognition results for all reference-characters with their respective scores.

[VB6]

Candidates As ()EOCR2CharacterCandidate

read-only

EOCR2Char.EOCR2Char

Constructs an EOCR2Char context.

[VB6]

```
void EOCR2Char()
)

void EOCR2Char(
    EOCR2Char other
)
```

Parameters

other

EOCR2Char object to be copied.

EOCR2Char.operator=

Copies all the data from another EOCR2Char object into the current EOCR2Char object

[VB6]

```
EOCR2Char operator=(  
    EOCR2Char other  
)
```

Parameters

other

EOCR2Char object to be copied

EOCR2Char.Text

Sets the value of the character, this is used during the learning phase.

[VB6]

Text As String

read-write

Remarks

It is also possible to set the character value using a **UINT16** code, this is done with [EOCR2Char::TextCode](#).

EOCR2Char.TextCode

Sets the value of the character, this is used during the learning phase.

[VB6]

TextCode As Long

read-write

Remarks

It is also possible to set the character value using a std::string, this is done with [SetText](#).

EOCR2Line Class

Holds a vector of [EOCR2Word](#) objects representing a line.

PROPERTIES

BoundingBox

Returns the bounding box encapsulating all characters in the line.

Text

Sets the value of all characters in the line with the text parameter.

Words

M Sets the vector of [EOCR2Word](#) representing the line.
E

THODS

[EOCR2Line](#)

Constructs an EOCR2Line context.

operator=

EOCR2Line.BoundingBox

Constructs an ERectAngle object from another EOCR2Line object.

Returns the bounding box encapsulating all characters in the line.

[VB6]

```
BoundingBox As ERectangle  
read-only
```

EOCR2Line.EOCR2Line

Constructs an EOCR2Line context.

[VB6]

```
void EOCR2Line()  
)  
void EOCR2Line(  
EOCR2Line other  
)
```

Parameters

other

EOCR2Line object to be copied.

EOCR2Line.operator=

Copies all the data from another EOCR2Line object into the current EOCR2Line object

[VB6]

```
EOCR2Line operator=(  
    EOCR2Line other  
)
```

Parameters

other

EOCR2Line object to be copied

EOCR2Line.Text

Sets the value of all characters in the line with the text parameter.

[VB6]

Text As String

read-write

Remarks

Use a space to separate two words.

EOCR2Line.Words

Sets the vector of [EOCR2Word](#) representing the line.

[VB6]

Words As ()EOCR2Word

read-write

EOCR2Text Class

Holds a vector of [EOCR2Line](#) objects representing a text.

PROPERTIES

BoundingBox

Returns the bounding box.encapsulating all characters in the text.

Lines

Sets the vector of [EOCR2Line](#) representing the text.

Text

M Sets the true value of all characters in the text with the text parameter.

E

THODS

EOCR2Text

Constructs an EOCR2Text context.

operator=

EOCR2Text.BoundingBox

Creates a Bounding Box from another EOCR2Text object.

Returns the bounding box.encapsulating all characters in the text.

[VB6]

BoundingBox As ERectangle

read-only

EOCR2Text.EOCR2Text

Constructs an EOCR2Text context.

[VB6]

```
void EOCR2Text()
)

void EOCR2Text(
    EOCR2Text other
)
```

Parameters

other

EOCR2Text object to be copied.

Remarks

Default and copy constructors.

EOCR2Text.Lines

Sets the vector of [EOCR2Line](#) representing the text.

[VB6]

Lines As ()EOCR2Line

read-write

EOCR2Text.operator=

Copies all the data from another EOCR2Text object into the current EOCR2Text object

[VB6]

```
EOCR2Text operator=(  
    EOCR2Text other  
)
```

Parameters

other

EOCR2Text object to be copied

EOCR2Text.Text

Sets the true value of all characters in the text with the text parameter.

[VB6]

Text As String

read-write

Remarks

Use a space (" ") to separate two words and a linebreak ("\n") to separate two lines.

EOCR2Word Class

Holds a vector of [EOCR2Char](#) objects representing a word.

PROPERTIES**BoundingBox**

Gets the bounding box of the word, encapsulating all characters in the word.

Characters

Sets the vector of [EOCR2Char](#) representing the word.

Text

M Sets the value of all characters in the word with the text parameter.

E

THODS**EOCR2Word**

Constructs an EOCR2Word context.

operator=

E Copies all the data from another EOCR2Word object into the current EOCR2Word object

O

CR2Word.BoundingBox

Gets the bounding box of the word, encapsulating all characters in the word.

[VB6]

BoundingBox As ERectangle

read-only

EOCR2Word.Characters

Sets the vector of [EOCR2Char](#) representing the word.

[VB6]

Characters As ()EOCR2Char

read-write

EOCR2Word.EOCR2Word

Constructs an EOCR2Word context.

[VB6]

```
void EOCR2Word()  
)  
  
void EOCR2Word(  
EOCR2Word other  
)
```

Parameters

other

EOCR2Word object to be copied.

EOCR2Word.operator=

Copies all the data from another EOCR2Word object into the current EOCR2Word object

[VB6]

```
EOCR2Word operator= (
    EOCR2Word other
)
```

Parameters

other

EOCR2Word object to be copied

EOCR2Word.Text

Sets the value of all characters in the word with the text parameter.

[VB6]

Text As String

read-write

EOCV Class

Manages a complete context for the optical character verification tool implemented in EasyOCV.

PROPERTIES

AccurateTextsLocationScores

Current texts location scoring mode.

ContrastAverageAverage contrast of last images added to statistics, or **FLOAT32_UNDEFINED** if it cannot be computed.

ContrastDeviation	Standard deviation of the contrast of the last images added to statistics, or FLOAT32_UNDEFINED if it cannot be computed.
ContrastTolerance	Allowed tolerance on the global contrast (allowed difference between the sample and template values).
Diagnostics	Logical combination (bitwise OR) of the defects found.
FreeCharCount	-
LocationMode	Kind of pre-processing should be applied to the sample image, as defined by ELocationMode .
NormalizeLocationScore	Current location score normalization mode.
NumTexts	Number of texts in the current OCV context.
ReduceLocationScore	Current location score reduction mode.
ResampleChars	Character resampling mode.
SampleBackground	Reference background gray levels determined during inspection of the sample.

[SampleContrast](#)

Global contrast of the last inspected image, or **FLOAT32_UNDEFINED** if it has not been computed.

[SampleForeground](#)

Reference foreground gray levels determined during inspection of the sample.

[SampleThreshold](#)

Threshold level applied to the sample during inspection.

[StatisticsCount](#)

Number of samples that have been taken into account in the statistics so far.

[TemplateBackground](#)

Reference background gray levels determined during learning of the template.

[TemplateContrast](#)

Global contrast of the template image, or **FLOAT32_UNDEFINED** if it has not been computed.

[TemplateForeground](#)

Reference foreground gray levels determined during learning of the template.

TemplateImage	Source template image associated to the OCV context during model edition.
TemplateThreshold	Threshold level applied to the template during learning.
UsedQualityIndicators	Choice of quality indicators.
WhiteOnBlack	M Flag indicating whether the learning and inspection steps use white characters on a black background, or black characters on a white background.
THODS	
AdjustCharsLocationRanges	Adjusts the location ranges (i.e tolerances and bias) of the characters using the results of the statistical training (minimum and maximum observed values).
AdjustCharsQualityRanges	Adjusts the allowed ranges for the character quality indicators (i.e. tolerances and bias) using the results of the statistical training (average and standard deviation of the observed values).
AdjustShiftTolerances	Adjust the shift tolerances on texts from a provided ROI.

[AdjustTextsLocationRanges](#)

Adjusts the location ranges (i.e tolerances and bias) of texts using the results of the statistical training (minimum and maximum observed values).

[AdjustTextsQualityRanges](#)

Adjusts the allowed ranges for the text quality indicators (i.e. tolerances and bias) using the results of the statistical training (average and standard deviation of the observed values).

[ClearStatistics](#)

Resets the statistics accumulation.

[ComputeDefaultTolerances](#)

Computes or recomputes the default tolerances of the quality indicators currently in use for all characters of all texts from the given ROI and threshold.

[CreateTemplateChars](#)

Adds to the OCV context the characters formed from the list of free objects.

[CreateTemplateObjects](#)

Adds to the OCV context the objects from the list of the associated coded image.

[CreateTemplateTexts](#)

Adds to the OCV context a piece of text formed from the list of free characters.

[DeleteTemplateChars](#)

Removes free characters from the OCV context.

DeleteTemplateObjects	Removes free objects from the OCV context.
DeleteTemplateTexts	Removes template texts from the OCV context.
DrawTemplateChars	Draws the free characters as bounding rectangles.
DrawTemplateCharsWithCurrentPen	Draws the free characters as bounding rectangles.
	DrawTemplateObjects
	Draws the free objects as blobs.

[DrawTemplateObjectsWithCurrentPen](#)

Draws the free objects as blobs.

[DrawTemplateTexts](#)

Draws the texts as bounding rectangles.

DrawTemplateTextsChars	Draws the characters of the texts as bounding rectangles.
DrawTemplateTextsCharsWithCurrentPen	Draws the characters of the texts as bounding rectangles.
DrawTemplateTextsWithCurrentPen	Draws the texts as bounding rectangles.
DrawText	
	Draws a tight bounding box around a single text, possibly with the main diagonal where diagnostics occur.

[DrawTextChars](#)

Draws a tight bounding box around all characters of a single text, possibly with the main diagonal where diagnostics occur.

[DrawTextCharsWithCurrentPen](#)

Draws a tight bounding box around all characters of a single text, possibly with the main diagonal where diagnostics occur.

[DrawTexts](#)

Draws a tight bounding box around all texts, possibly with the main diagonal where diagnostics occur.

[DrawTextsChars](#)

Draws a tight bounding box around all characters of all texts.

[DrawTextsCharsWithCurrentPen](#)

Draws a tight bounding box around all characters of all texts.

[DrawTextsWithCurrentPen](#)

Draws a tight bounding box around all texts, possibly with the main diagonal where diagnostics occur.

[DrawTextWithCurrentPen](#)

Draws a tight bounding box around a single text, possibly with the main diagonal where diagnostics occur.

EOCV	Constructs an OCV context.
GatherTextsCharsParameters	Copies the parameters from the characters specified by the selection modes to the temporary character.
GatherTextsParameters	Copies the parameters from the texts specified by the selection mode to the temporary text.
GetFreeChar	Gets an individual free character of the OCV context.
GetNumTextChars	Returns the number of characters in this OCV context.
GetText	Returns an individual text of the OCV context.
GetTextCharParameters	Copies the parameters from the character of given index to the temporary character.
GetTextCharPoint	Computes the coordinates of a point located relatively to a given character.
GetTextParameters	Copies the parameters from the text of given index to the temporary text.
GetTextPoint	Computes the coordinates of a point located relatively to a given text.

Inspect

Inspection is the process that locates the template in the sample image using all allowed degrees of freedom, compares both images to detects and quantify defects.

Learn

Pre-processes the model so that all required internal data structures are set up.

Load

-

Save

-

ScatterTextsCharsParameters

Copies the desired parameters from the temporary character to the characters specified by the selection mode.

ScatterTextsParameters

Copies the desired parameters from the temporary text to the texts specified by the selection mode.

SelectSampleTexts

Toggles selection of the template texts whose bounding box intersects a given rectangle.

SelectSampleTextsChars

Toggles selection of the template text characters whose bounding box intersects a given rectangle.

SelectTemplateChars

Toggles selection of the free characters whose bounding box intersects a given rectangle.

[SelectTemplateObjects](#)

Toggles selection of the objects whose bounding box intersects a given rectangle.

[SelectTemplateTexts](#)

Toggles selection of the template texts whose bounding box intersects a given rectangle.

[SetFreeChar](#)

Sets an individual free character of the OCV context.

[SetText](#)

Modifies the individual text at the given index by the given value.

[SetTextCharParameters](#)

Copies the desired parameters from the temporary character to the character of given index.

[SetTextParameters](#)

Copies the desired parameters from the temporary text to the text of given index.

[UpdateStatistics](#)

Adds the parameters of the last inspection to the statistics.

CV.AccurateTextsLocationScores

Current texts location scoring mode.

[VB6]

```
AccurateTextsLocationScores As Boolean
```

read-write

Remarks

During text location, the text location score is computed before any individual character displacement is allowed. This results in lower location scores because the matching between undeformed template and sample texts is less accurate. When turned on, this property allows the score to be corrected, by taking into account the character displacements. This allows more robust diagnostics based on the location score. If no movement freedom is given to the characters (no **ShiftTolerance**), this option is irrelevant.

EOCV.AdjustCharsLocationRanges

Adjusts the location ranges (i.e tolerances and bias) of the characters using the results of the statistical training (minimum and maximum observed values).

[VB6]

```
void AdjustCharsLocationRanges (
    Single factor,
    ESelectionFlag textSelection,
    ESelectionFlag charSelection
)
```

Parameters

factor

Safety factor to use when re-computing the tolerances.

textSelection

Tells on which texts the adjustment should be applied, as defined by [ESelectionFlag](#).

charSelection

Tells on which chars the adjustment should be applied, as defined by [ESelectionFlag](#).

Remarks

The method reassigns bias and tolerances values according to their minimum and maximum observed values in the X and Y directions (reported by statistics). The following adjustments are made on characters whose selection state matches **textSelection** and **charSelection**: * **Shift Bias** is assigned $1/2 * (\text{Shift Max} + \text{Shift Min})$ * **Shift Tolerance** is assigned $1/2 * \text{factor} * (\text{Shift Max} - \text{Shift Min})$ Defined this way, the allowed range is simply the observed range enlarged by the user-defined factor. The default value for **factor** is **1.2** (+20 % allowance).

EOCV.AdjustCharsQualityRanges

Adjusts the allowed ranges for the character quality indicators (i.e. tolerances and bias) using the results of the statistical training (average and standard deviation of the observed values).

[VB6]

```
void AdjustCharsQualityRanges (
    Single factor,
    ESelectionFlag textSelection,
    ESelectionFlag charSelection
)
```

Parameters

factor

Safety factor to use when re-computing the tolerances.

textSelection

Tells on which texts the adjustment should be applied, as defined by [ESelectionFlag](#).

charSelection

Tells on which chars the adjustment should be applied, as defined by [ESelectionFlag](#).

Remarks

The method reassigns bias and tolerances values according to their average and standard deviation, as reported by statistics. The following adjustments are made on characters whose selection state matches **textSelection** and **charSelection**: * the bias parameter value is assigned the corresponding parameter average * the tolerance parameter value is assigned the product of **factor** by its corresponding standard deviation Please note that **3** is a minimum value for **factor** (3sigma rule).This value should be increased if deviations are computed from a small number of samples.

EOCV.AdjustShiftTolerances

Adjust the shift tolerances on texts from a provided ROI.

[VB6]

```
void AdjustShiftTolerances(
    EROIBW8 roi
)
```

Parameters

roi

Pointer on the ROI from which the shift tolerances will be computed.

Remarks

The method performs the computation of texts shift tolerances, assuming that the specified ROI always contains the whole text (text cannot lie outside of the ROI). Other text location parameters (skew, scale, shear) are not taken in account, and this method should not be called if these parameters tolerances are not set to zero. The method also assumes texts always have the same displacement, and does not affect characters location tolerances.

EOCV.AdjustTextsLocationRanges

Adjusts the location ranges (i.e tolerances and bias) of texts using the results of the statistical training (minimum and maximum observed values).

[VB6]

```
void AdjustTextsLocationRanges (
    Single factor,
    ESelectionFlag selection
)
```

Parameters

factor

Safety factor to use when re-computing the tolerances.

selection

Tells on which texts the adjustment should be applied, as defined by [ESelectionFlag](#).

Remarks

The method reassigns bias and tolerances values according to their minimum and maximum observed values in the X and Y directions (reported by statistics). The following adjustments are made on texts whose selection state matches **selection**: * **Shift Bias** is assigned **1/2** * **(Shift Max + Shift Min)** * **Shift Tolerance** is assigned **1/2** * **factor** * **(Shift Max - Shift Min)**Defined this way, the allowed range is simply the observed range enlarged by the user-defined factor. The default value for **factor** is **1.2** (+20 % allowance).

EOCV.AdjustTextsQualityRanges

Adjusts the allowed ranges for the text quality indicators (i.e. tolerances and bias) using the results of the statistical training (average and standard deviation of the observed values).

[VB6]

```
void AdjustTextsQualityRanges (
    Single factor,
    ESelectionFlag selection
)
```

Parameters

factor

Safety factor to use when re-computing the tolerances.

selection

Tells on which texts the adjustment should be applied, as defined by [ESelectionFlag](#).

Remarks

The method reassigns bias and tolerances values according to their average and standard deviation, as reported by statistics. The following adjustments are made on texts whose selection state matches **selection**: * the bias parameter value is assigned the corresponding parameter average * the tolerance parameter value is assigned the product of **factor** by its corresponding standard deviation Please note that **3** is a minimum value for **factor** (3sigma rule).This value should be increased if deviations are computed from a small number of samples.

EOCV.ClearStatistics

Resets the statistics accumulation.

```
[VB6]  
void ClearStatistics()  
)
```

Remarks

This function must be called before a batch of inspections.

EOCV.ComputeDefaultTolerances

Computes or recomputes the default tolerances of the quality indicators currently in use for all characters of all texts from the given ROI and threshold.

```
[VB6]  
void ComputeDefaultTolerances(  
    EROIBW8 roi,  
    Long threshold  
)
```

Parameters

roi

Pointer to the ROI from which the default tolerances should be computed.

threshold

Threshold level to use during computation, as defined by [EThresholdMode](#).

Remarks

An explicit call of this method is performed by method Learn at the end of the learning stage.

EOCV.CongtrastAverage

Average contrast of last images added to statistics, or **FLOAT32_UNDEFINED** if it cannot be computed.

[VB6]

ContrastAverage As Single

read-only

Remarks

The contrast is defined as the ratio of the reference gray-levels difference over their sum. This parameter is maximum (100 %) for a perfectly contrasted image (white on black).

EOCV.CongtrastDeviation

Standard deviation of the contrast of the last images added to statistics, or **FLOAT32_UNDEFINED** if it cannot be computed.

[VB6]

ContrastDeviation As Single

read-only

Remarks

The contrast is defined as the ratio of the reference gray-levels difference over their sum. This parameter is maximum (100 %) for a perfectly contrasted image (white on black).

EOCV.CongtrastTolerance

Allowed tolerance on the global contrast (allowed difference between the sample and template values).

[VB6]

ContrastTolerance As Single

read-write

Remarks

The contrast is defined as the ratio of the reference gray-levels difference over their sum. This parameter is maximum (100 %) for a perfectly contrasted image (white on black).

EOCV.CreateTemplateChars

Adds to the OCV context the characters formed from the list of free objects.

[VB6]

```
void CreateTemplateChars(
    ESelectionFlag objectsSelectionFlag,
    ECharCreationMode creationMode
)
```

Parameters

objectsSelectionFlag

Selection mode for the objects, as defined by [ESelectionFlag](#). By default, only the selected objects are created.

creationMode

Free objects grouping criterion, as defined by [ECharCreationMode](#). By default, the free objects whose bounding rectangle overlap are considered as belonging to the same character.

Remarks

The free characters are sets of blobs taken from the free objects list. One step of the model definition is to specify which free objects will be handled by the OCV context and how they will be grouped to form characters. Grouping can be done automatically by using a overlapping criterion, or explicitly. When an object is added to a character, it is removed from the free objects list.

EOCV.CreateTemplateObjects

Adds to the OCV context the objects from the list of the associated coded image.

[VB6]

```
void CreateTemplateObjects(
    ECodedImage codedImage,
    ESelectionFlag codedObjectsSelectionFlag
)
```

Parameters

codedImage

An **ECodedImage** object.

codedObjectsSelectionFlag

Selection mode for the objects, as defined by **ESelectionFlag**. By default, only the selected objects are created.

Remarks

The free objects are blobs built using a separate coded image. One step of the model definition is to specify which objects will be handled by the OCV context.

EOCV.CreateTemplateTexts

Adds to the OCV context a piece of text formed from the list of free characters.

[VB6]

```
void CreateTemplateTexts(
    ESelectionFlag charsSelectionFlag
)
```

Parameters

charsSelectionFlag

Selection mode for the free characters, as defined by [ESelectionFlag](#). By default, only the selected free characters are processed.

Remarks

The template texts are sets of characters taken from the free characters list. One step of the model definition is to specify which free characters will be handled by the OCV context and how they will be grouped to form texts. Grouping must be done explicitly. When a character is added to a text, it is removed from the free characters list.

EOCV.DeleteTemplateChars

Removes free characters from the OCV context.

[VB6]

```
void DeleteTemplateChars(
    ESelectionFlag charsSelectionFlag
)
```

Parameters

charsSelectionFlag

Selection mode for the free characters, as defined by [ESelectionFlag](#). By default, only the selected free characters are deleted.

Remarks

The free characters are sets of blobs taken from the free objects list. One step of the model definition is to specify which free objects will be handled by the OCV context and how they will be grouped to form characters. Grouping can be done automatically by using a overlapping criterion, or explicitly. When an object is added to a character, it is removed from the free objects list.

EOCV.DeleteTemplateObjects

Removes free objects from the OCV context.

[VB6]

```
void DeleteTemplateObjects(
    ESelectionFlag objectsSelectionFlag
)
```

Parameters

objectsSelectionFlag

Selection mode for the objects, as defined by [ESelectionFlag](#). By default, only the selected objects are deleted.

Remarks

The free objects are blobs built using a separate coded image. One step of the model definition is to specify which objects will be handled by the OCV context.

EOCV.DeleteTemplateTexts

Removes template texts from the OCV context.

[VB6]

```
void DeleteTemplateTexts(
    ESelectionFlag textsSelectionFlag
)
```

Parameters

textsSelectionFlag

Selection mode for the template texts, as defined by [ESelectionFlag](#). By default, only the selected texts are deleted.

Remarks

The template texts are sets of characters taken from the free characters list. One step of the model definition is to specify which free characters will be handled by the OCV context and how they will be grouped to form texts. Grouping must be done explicitly. When a character is added to a text, it is removed from the free characters list.

EOCV.Diagnostics

Logical combination (bitwise OR) of the defects found.

[VB6]

Diagnostics As Long

read-only

Remarks

Inspection is the process that locates the template in the sample image, using all allowed degrees of freedom, and that compares both images to detect and quantify defects. After inspection, all defective texts and text characters are selected. Further, the diagnostics binary mask contain a detailed interpretation of the defects found.

EOCV.DrawTemplateChars

Draws the free characters as bounding rectangles.

[VB6]

```
void DrawTemplateChars(
    Long graphicContext,
    ESelectionFlag charsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

```

void DrawTemplateChars(
    Long graphicContext,
    ERGBColor color,
    ESelectionFlag charsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

void DrawTemplateChars(
    EDrawAdapter graphicContext,
    ESelectionFlag charsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

```

Parameters

graphicContext

Device context of the drawing window.

charsSelectionFlag

Selection mode for the free characters, as defined by [ESelectionFlag](#). By default, only the selected characters are drawn.

zoomX

Zooming parameters.

zoomY

Zooming parameters.

originX

Panning parameters.

originY

Panning parameters.

color

-

Remarks

The template drawing methods are used to graphically represent the free objects, free characters, texts and text characters associated to the OCV context before learning. All drawing operations are done using the current pen attributes.

EOCV.DrawTemplateCharsWithCurrentPen

Draws the free characters as bounding rectangles.

[VB6]

```
void DrawTemplateCharsWithCurrentPen (
    Long graphicContext,
    ESelectionFlag charsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Device context of the drawing window.

charsSelectionFlag

Selection mode for the free characters, as defined by [ESelectionFlag](#). By default, only the selected characters are drawn.

zoomX

Zooming parameters.

zoomY

Zooming parameters.

originX

Panning parameters.

originY

Panning parameters.

Remarks

The template drawing methods are used to graphically represent the free objects, free characters, texts and text characters associated to the OCV context before learning. All drawing operations are done using the current pen attributes.

EOCV.DrawTemplateObjects

Draws the free objects as blobs.

[VB6]

```
void DrawTemplateObjects(
    Long graphicContext,
    ESelectionFlag objectsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

void DrawTemplateObjects(
    Long graphicContext,
    ERGBColor color,
    ESelectionFlag objectsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

void DrawTemplateObjects(
    EDrawAdapter graphicContext,
    ESelectionFlag objectsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Device context of the drawing window.

objectsSelectionFlag

Selection mode for the objects, as defined by [ESelectionFlag](#). By default, only the selected objects are drawn.

zoomX

Zooming parameters.

zoomY
 Zooming parameters.
originX
 Panning parameters.
originY
 Panning parameters.
color
 The color in which to draw the overlay.

Remarks

The associated coded image must be present. The template drawing methods are used to graphically represent the free objects, free characters, texts and text characters associated to the OCV context before learning. All drawing operations are done using the current pen attributes.

EOCV.DrawTemplateObjectsWithCurrentPen

Draws the free objects as blobs.

[VB6]

```
void DrawTemplateObjectsWithCurrentPen(
  Long graphicContext,
  ESelectionFlag objectsSelectionFlag,
  Single zoomX,
  Single zoomY,
  Single originX,
  Single originY
)
```

Parameters

graphicContext

Device context of the drawing window.

objectsSelectionFlag

Selection mode for the objects, as defined by [ESelectionFlag](#). By default, only the selected objects are drawn.

zoomX

Zooming parameters.

zoomY

Zooming parameters.

originX

Panning parameters.

originY

Panning parameters.

Remarks

The associated coded image must be present. The template drawing methods are used to graphically represent the free objects, free characters, texts and text characters associated to the OCV context before learning. All drawing operations are done using the current pen attributes.

EOCV.DrawTemplateTexts

Draws the texts as bounding rectangles.

[VB6]

```
void DrawTemplateTexts(
    Long graphicContext,
    ESelectionFlag textsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

void DrawTemplateTexts(
    Long graphicContext,
    ERGBColor color,
    ESelectionFlag textsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

```
void DrawTemplateTexts(
    EDrawAdapter graphicContext,
    ESelectionFlag textsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Device context of the drawing window.

textsSelectionFlag

Selection mode for the free texts, as defined by [ESelectionFlag](#). By default, only the selected texts are drawn.

zoomX

Zooming parameters.

zoomY

Zooming parameters.

originX

Panning parameters.

originY

Panning parameters.

color

The color in which to draw the overlay.

Remarks

The template drawing methods are used to graphically represent the free objects, free characters, texts and text characters associated to the OCV context before learning. All drawing operations are done using the current pen attributes.

EOCV.DrawTemplateTextsChars

Draws the characters of the texts as bounding rectangles.

[VB6]

```

void DrawTemplateTextsChars (
    Long graphicContext,
    ESelectionFlag textsSelectionFlag,
    ESelectionFlag charsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

void DrawTemplateTextsChars (
    Long graphicContext,
    ERGBColor color,
    ESelectionFlag textsSelectionFlag,
    ESelectionFlag charsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

void DrawTemplateTextsChars (
    EDrawAdapter graphicContext,
    ESelectionFlag textsSelectionFlag,
    ESelectionFlag charsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

```

Parameters

graphicContext

Device context of the drawing window.

textsSelectionFlag

Selection mode for the texts, as defined by [ESelectionFlag](#). By default, only characters from the selected texts are drawn.

charsSelectionFlag

Selection mode for the characters, as defined by [ESelectionFlag](#). By default, only the selected characters of the selected texts are drawn.

zoomX

Zooming parameters.

zoomY

Zooming parameters.

originX

Panning parameters.

originY

Panning parameters.

color

The color in which to draw the overlay.

Remarks

The template drawing methods are used to graphically represent the free objects, free characters, texts and text characters associated to the OCV context before learning. All drawing operations are done using the current pen attributes.

EOCV.DrawTemplateTextsCharsWithCurrentPen

Draws the characters of the texts as bounding rectangles.

[VB6]

```
void DrawTemplateTextsCharsWithCurrentPen(
    Long graphicContext,
    ESelectionFlag textsSelectionFlag,
    ESelectionFlag charsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Device context of the drawing window.

textsSelectionFlag

Selection mode for the texts, as defined by [ESelectionFlag](#). By default, only characters from the selected texts are drawn.

charsSelectionFlag

Selection mode for the characters, as defined by [ESelectionFlag](#). By default, only the selected characters of the selected texts are drawn.

zoomX

Zooming parameters.

zoomY

Zooming parameters.

originX

Panning parameters.

originY

Panning parameters.

Remarks

The template drawing methods are used to graphically represent the free objects, free characters, texts and text characters associated to the OCV context before learning. All drawing operations are done using the current pen attributes.

EOCV.DrawTemplateTextsWithCurrentPen

Draws the texts as bounding rectangles.

[VB6]

```
void DrawTemplateTextsWithCurrentPen(
    Long graphicContext,
    ESelectionFlag textsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

Parameters*graphicContext*

Device context of the drawing window.

*textsSelectionFlag*Selection mode for the free texts, as defined by [ESelectionFlag](#). By default, only the selected texts are drawn.*zoomX*

Zooming parameters.

zoomY

Zooming parameters.

originX

Panning parameters.

originY

Panning parameters.

Remarks

The template drawing methods are used to graphically represent the free objects, free characters, texts and text characters associated to the OCV context before learning. All drawing operations are done using the current pen attributes.

EOCV.DrawText

Draws a tight bounding box around a single text, possibly with the main diagonal where diagnostics occur.

```
[VB6]

void DrawText(
    Long graphicContext,
    EOCVText text,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

void DrawText(
    Long graphicContext,
    ERGBColor color,
    EOCVText text,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

void DrawText(
    EDrawAdapter graphicContext,
    EOCVText text,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

Parameters*graphicContext*

Device context of the drawing window.

*text*Reference to the desired text can be obtained by using [EOCV::GetText](#).*zoomX*

Zooming parameters.

zoomY

Zooming parameters.

originX

Panning parameters.

originY

Panning parameters.

color

The color in which to draw the overlay.

Remarks

The sample drawing member functions are used to graphically represent the texts and text characters located in the image during inspection. All drawing operations are done using the current pen attributes. Normally, the characters and texts are drawn as a bounding box. However, when diagnostics have been raised on a character or text, the main diagonal (north-west to south-east) is drawn as well, indicating that the item is rejected.

EOCV.DrawTextChars

Draws a tight bounding box around all characters of a single text, possibly with the main diagonal where diagnostics occur.

[VB6]

```
void DrawTextChars(
    Long graphicContext,
    EOCVText text,
    ESelectionFlag charsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

```
void DrawTextChars(
    Long graphicContext,
    ERGBColor color,
    EOCVText text,
    ESelectionFlag charsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

void DrawTextChars(
    EDrawAdapter graphicContext,
    EOCVText text,
    ESelectionFlag charsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Handle to the device context of the destination window.

text

Reference to the desired text.

charsSelectionFlag

Characters selection mode, as defined by [ESelectionFlag](#).

zoomX

Zooming parameters.

zoomY

Zooming parameters.

originX

Panning parameters.

originY

Panning parameters.

color

The color in which to draw the overlay.

Remarks

The sample drawing methods are used to graphically represent the texts and text characters located in the image during inspection. All drawing operations are done using the current pen attributes. Normally, the characters and texts are drawn as a bounding box. However, when diagnostics have been raised on a character or text, the main diagonal (north west to south east) is drawn as well, indicating that the item is rejected.

EOCV.DrawTextCharsWithCurrentPen

Draws a tight bounding box around all characters of a single text, possibly with the main diagonal where diagnostics occur.

[VB6]

```
void DrawTextCharsWithCurrentPen(
    Long graphicContext,
    EOCVText text,
    ESelectionFlag charsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Handle to the device context of the destination window.

text

Reference to the desired text.

charsSelectionFlag

Characters selection mode, as defined by [ESelectionFlag](#).

zoomX

Zooming parameters.

zoomY

Zooming parameters.

originX

Panning parameters.

originY

Panning parameters.

Remarks

The sample drawing methods are used to graphically represent the texts and text characters located in the image during inspection. All drawing operations are done using the current pen attributes. Normally, the characters and texts are drawn as a bounding box. However, when diagnostics have been raised on a character or text, the main diagonal (north west to south east) is drawn as well, indicating that the item is rejected.

EOCV.DrawTexts

Draws a tight bounding box around all texts, possibly with the main diagonal where diagnostics occur.

[VB6]

```
void DrawTexts(
    Long graphicContext,
    ESelectionFlag textsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

void DrawTexts(
    Long graphicContext,
    ERGBColor color,
    ESelectionFlag textsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

void DrawTexts(
    EDrawAdapter graphicContext,
    ESelectionFlag textsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Device context of the drawing window.

textsSelectionFlag

Texts selection mode, as defined by [ESelectionFlag](#).

zoomX

Zooming parameters.

zoomY

Zooming parameters.

originX

Panning parameters.

originY

Panning parameters.

color

The color in which to draw the overlay.

Remarks

The sample drawing methods are used to graphically represent the texts and text characters located in the image during inspection. All drawing operations are done using the current pen attributes. Normally, the characters and texts are drawn as a bounding box. However, when diagnostics have been raised on a character or text, the main diagonal (north west to south east) is drawn as well, indicating that the item is rejected.

EOCV.DrawTextsChars

Draws a tight bounding box around all characters of all texts.

[VB6]

```
void DrawTextsChars (
    Long graphicContext,
    ESelectionFlag textsSelectionFlag,
    ESelectionFlag charsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

```
void DrawTextsChars(
    Long graphicContext,
    ERGBColor color,
    ESelectionFlag textsSelectionFlag,
    ESelectionFlag charsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

void DrawTextsChars(
    EDRApapter graphicContext,
    ESelectionFlag textsSelectionFlag,
    ESelectionFlag charsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Device context of the drawing window.

textsSelectionFlag

Texts selection mode, as defined by [ESelectionFlag](#).

charsSelectionFlag

Characters selection mode, as defined by [ESelectionFlag](#).

zoomX

Zooming parameters.

zoomY

Zooming parameters.

originX

Panning parameters.

originY

Panning parameters.

color

The color in which to draw the overlay.

Remarks

The sample drawing methods are used to graphically represent the texts and text characters located in the image during inspection. All drawing operations are done using the current pen attributes. Normally, the characters and texts are drawn as a bounding box. However, when diagnostics have been raised on a character or text, the main diagonal (north west to south east) is drawn as well, indicating that the item is rejected.

EOCV.DrawTextsCharsWithCurrentPen

Draws a tight bounding box around all characters of all texts.

[VB6]

```
void DrawTextsCharsWithCurrentPen(
    Long graphicContext,
    ESelectionFlag textsSelectionFlag,
    ESelectionFlag charsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Device context of the drawing window.

textsSelectionFlag

Texts selection mode, as defined by [ESelectionFlag](#).

charsSelectionFlag

Characters selection mode, as defined by [ESelectionFlag](#).

zoomX

Zooming parameters.

zoomY

Zooming parameters.

originX

Panning parameters.

originY

Panning parameters.

Remarks

The sample drawing methods are used to graphically represent the texts and text characters located in the image during inspection. All drawing operations are done using the current pen attributes. Normally, the characters and texts are drawn as a bounding box. However, when diagnostics have been raised on a character or text, the main diagonal (north west to south east) is drawn as well, indicating that the item is rejected.

EOCV.DrawTextsWithCurrentPen

Draws a tight bounding box around all texts, possibly with the main diagonal where diagnostics occur.

[VB6]

```
void DrawTextsWithCurrentPen (
    Long graphicContext,
    ESelectionFlag textsSelectionFlag,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Device context of the drawing window.

textsSelectionFlag

Texts selection mode, as defined by [ESelectionFlag](#).

zoomX

Zooming parameters.

zoomY

Zooming parameters.

originX

Panning parameters.

originY

Panning parameters.

Remarks

The sample drawing methods are used to graphically represent the texts and text characters located in the image during inspection. All drawing operations are done using the current pen attributes. Normally, the characters and texts are drawn as a bounding box. However, when diagnostics have been raised on a character or text, the main diagonal (north west to south east) is drawn as well, indicating that the item is rejected.

EOCV.DrawTextWithCurrentPen

Draws a tight bounding box around a single text, possibly with the main diagonal where diagnostics occur.

[VB6]

```
void DrawTextWithCurrentPen(
    Long graphicContext,
    EOCVText text,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Device context of the drawing window.

text

Reference to the desired text can be obtained by using [EOCV::GetText](#).

zoomX

Zooming parameters.

zoomY

Zooming parameters.

originX

Panning parameters.

originY

Panning parameters.

Remarks

The sample drawing member functions are used to graphically represent the texts and text characters located in the image during inspection. All drawing operations are done using the current pen attributes. Normally, the characters and texts are drawn as a bounding box. However, when diagnostics have been raised on a character or text, the main diagonal (north-west to south-east) is drawn as well, indicating that the item is rejected.

EOCV.EOCV

Constructs an OCV context.

```
[VB6]  
void EOCV(  
    EOCV other  
)  
  
void EOCV(  
)
```

Parameters

other

Remarks

Only the default constructor is needed.

EOCV.FreeCharCount

-

```
[VB6]  
FreeCharCount As Long  
read-only
```

EOCV.GatherTextsCharsParameters

Copies the parameters from the characters specified by the selection modes to the temporary character.

[VB6]

```
void GatherTextsCharsParameters(
    EOCVChar ocvChar,
    ESelectionFlag textsSelectionFlag,
    ESelectionFlag charsSelectionFlag
)
```

Parameters

ocvChar

Temporary [EOCVChar](#) object.

textsSelectionFlag

Selection mode for the texts, as defined by [ESelectionFlag](#).

charsSelectionFlag

Selection mode for the text characters, as defined by [ESelectionFlag](#).

Remarks

For each character parameter that has a unique value among the specified characters, the corresponding value is different from undefined. Always be sure to check that a returned parameter does not have such an undefined value assigned, otherwise its contents will appear meaningless.

EOCV.GatherTextsParameters

Copies the parameters from the texts specified by the selection mode to the temporary text.

[VB6]

```
void GatherTextsParameters(
    EOCVText Text,
    ESelectionFlag textsSelectionFlag
)
```

Parameters

Text

Selection mode for the texts, as defined by [ESelectionFlag](#).

textsSelectionFlag

Temporary [EOCVText](#) object.

Remarks

For each text parameter that has a unique value among the specified texts, the corresponding value is different from undefined. Always be sure to check that a returned parameter does not have such an undefined value assigned, otherwise its contents will appear meaningless.

EOCV.GetFreeChar

Gets an individual free character of the OCV context.

[VB6]

```
EOCVChar GetFreeChar(
    Long index
)
```

Parameters

index

Index of the individual free character.

Remarks

The free characters are sets of blobs taken from the free objects list. One step of the model definition is to specify which free objects will be handled by the OCV context, and how they will be grouped to form characters. Grouping can be done automatically by using a overlapping criterion, or explicitly. When an object is added to a character, it is removed from the free objects list.

EOCV.GetNumTextChars

Returns the number of characters in this OCV context.

[VB6]

```
Long GetNumTextChars(  
    Long index  
)
```

Parameters

index

Index of the desired text, in range **0** to **NumTexts-1**.

Remarks

The characters are numbered from **0** to **NumTextChars** excluded.

EOCV.GetText

Returns an individual text of the OCV context.

[VB6]

```
EOCVText GetText(  
    Long index  
)
```

Parameters

index

Index, between **0** and **EOCV::NumTexts** (excluded) of the text to be accessed.

EOCV.GetTextCharParameters

Copies the parameters from the character of given index to the temporary character.

```
[VB6]  
void GetTextCharParameters(  
    EOCVChar ocvChar,  
    Long textIndex,  
    Long charIndex  
)
```

Parameters

ocvChar

Temporary [EOCVChar](#) object.

textIndex

Character index in range **0** to **NumTexts-1**.

charIndex

Character index in range **0** to **NumTextChars-1**.

Remarks

For each character parameter retrieved, the corresponding value is different from undefined. Always be sure to check that a returned parameter does not have such an undefined value assigned, otherwise its contents will appear meaningless.

EOCV.GetTextCharPoint

Computes the coordinates of a point located relatively to a given character.

```
[VB6]
```

```
void GetTextCharPoint(
    Long textIndex,
    Long charIndex,
    Long x,
    Long y,
    Single reducedX,
    Single reducedY,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

textIndex

Text index in range **0..NumTexts**, excluded.

charIndex

Character index in range **0..NumTextChars**, excluded.

x

returned abscissa of the requested point.

y

returned ordinate of the requested point.

reducedX

X position of the requested point relatively to the character bounding box, as defined on the picture below.

reducedY

Y position of the requested point relatively to the character bounding box, as defined on the picture below

zoomX

Zooming parameters.

zoomY

Zooming parameters.

panX

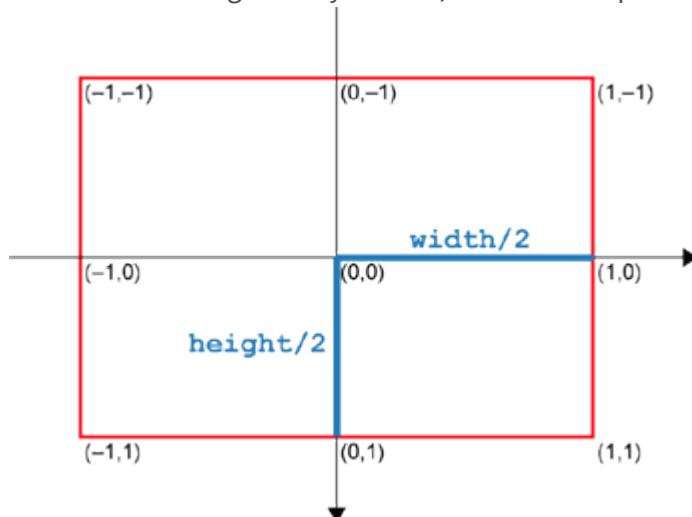
Panning parameters.

panY

Panning parameters.

Remarks

The parameters **reducedX** and **reducedY** are used to indicate the position of the requested point relatively to the bounding box. These parameters can take values from **-1** to **1**, where **1** is the half height or half width of the bounding box. By default, the returned point is the



center of the bounding box.

EOCV.GetTextParameters

Copies the parameters from the text of given index to the temporary text.

[VB6]

```
void GetTextParameters(
    EOCVText Text,
    Long textIndex
)
```

Parameters

Text

Text index in range **0** to **NumTexts**, excluded.

textIndex

Text index in range **0** to **NumTexts**, excluded.

Remarks

For each text parameter retrieved, the corresponding value is different from undefined. Always be sure to check that a returned parameter does not have such an undefined value assigned, otherwise its contents will appear meaningless.

EOCV.GetTextPoint

Computes the coordinates of a point located relatively to a given text.

[VB6]

```
void GetTextPoint(  
    Long textIndex,  
    Long x,  
    Long y,  
    Single reducedX,  
    Single reducedY,  
    Single zoomX,  
    Single zoomY,  
    Single panX,  
    Single panY  
)
```

Parameters

textIndex

Text index in range **0..NumTexts**, excluded.

x

Returned abscissa of the requested point.

y

Returned ordinate of the requested point.

reducedX

X position of the requested point relatively the text bounding box, as defined on the picture below.

reducedY

Y position of the requested point relatively to the text bounding box, as defined on the picture below.

zoomX

Zooming parameters.

zoomY

Zooming parameters.

panX

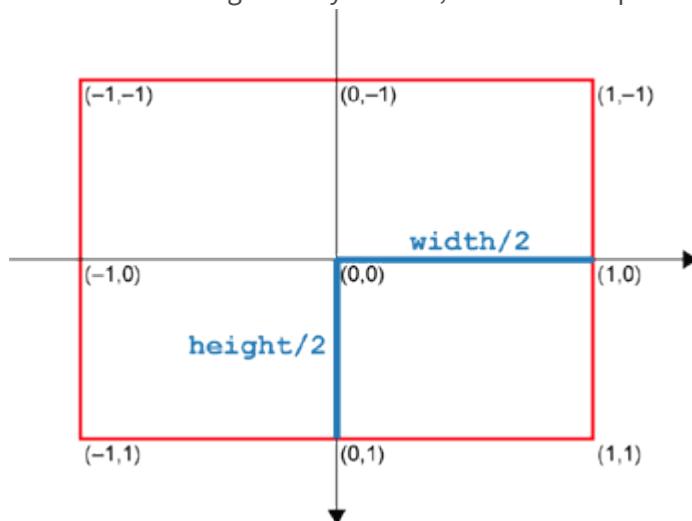
Panning parameters.

panY

Panning parameters.

Remarks

The parameters **reducedX** and **reducedY** are used to indicate the position of the requested point relatively to the bounding box. These parameters can take values from **-1** to **1**, where **1** is the half height or half width of the bounding box. By default, the returned point is the



center of the bounding box.

EOCV.Inspect

Inspection is the process that locates the template in the sample image using all allowed degrees of freedom, compares both images to detect and quantify defects.

```
[VB6]
void Inspect(
    EROIBW8 sampleImage,
    Long threshold
)
```

Parameters

sampleImage

Pointer to the image/ROI to be inspected. The center of this image serves as a reference point on which the center of the template image is aligned.

threshold

Threshold level suitable for the sample image. The same convention as that of **ImgThreshold** is used for automatic thresholding.

Remarks

After inspection, all defective texts and text characters are selected. Further, the Diagnostics binary mask contain a detailed interpretation of the defects found. A few options can be tuned to speed up the inspection process (see Location Options and Inspection Options).

EOCV.Learn

Pre-processes the model so that all required internal data structures are set up.

[VB6]

```
void Learn(
    EROIBW8 image,
    ESelectionFlag textsSelectionFlag,
    ESelectionFlag charsSelectionFlag
)
```

Parameters

image

Pointer to the template image/ROI.

textsSelectionFlag

Texts selection mode, as defined by [ESelectionFlag](#).

charsSelectionFlag

Characters selection mode, as defined by [ESelectionFlag](#).

Remarks

After learning, the model may not be edited anymore. It can be saved for later use and becomes ready for the inspection task. Learning is the final step of the model editing.

EOCV.Load

-

[VB6]

```
void Load(  
    ESerializer serializer  
)  
  
void Load(  
    String stream  
)
```

Parameters

serializer

-

stream

-

EOCV.LocationMode

Kind of pre-processing should be applied to the sample image, as defined by [ELocationMode](#).

[VB6]

LocationMode As ELocationMode

read-write

Remarks

The location process is an important phase of the inspection task. It allows the OCV context to accurately find the template position in the sample image, using all allowed degrees of freedom. This process can be tuned in several ways to trade speed for robustness.

EOCV.NormalizeLocationScore

Current location score normalization mode.

[VB6]

NormalizeLocationScore As Boolean

read-write

Remarks

Location score normalization performs a score correction, using the sample and template reference gray levels. It is intended to correct potential contrast or intensity discrepancies between template and sample images. It acts as if the sample image had the same foreground and background reference gray levels than the template image. It is recommended to turn this option on if the acquisition conditions can change from one sample to another. However, the correction may have a bad effect if the image acquisition system (camera) uses gamma correction, or the acquired image is saturated. The *location score normalization* mode is independent of the *location score reduction* mode.

EOCV.NumTexts

Number of texts in the current OCV context.

[VB6]

NumTexts As Long

read-only

Remarks

The texts are numbered from **0** to **NumTexts-1**.

EOCV.ReduceLocationScore

Current location score reduction mode.

[VB6]

ReduceLocationScore As Boolean

read-write

Remarks

The location score is computed as a sum of gray-level values along the character contours. When the reduction mode is enabled (default), the location score is divided by the number of contour points, giving an average gray level, in range **0..255**. The non-reduction mode is considered obsolete. The *location score reduction* mode is independent of the *location score normalization* mode.

EOCV.ResampleChars

Character resampling mode.

[VB6]

ResampleChars As Boolean

read-write

Remarks

Normally, when text is inspected with rotation, scaling and/or shearing, some resampling must be performed when computing the quality indicators on the separate characters. If the angles remain small and the scale factors remain very close to unity, this resampling can be avoided by setting this property to **FALSE**.

EOCV.SampleBackground

Reference background gray levels determined during inspection of the sample.

[VB6]

SampleBackground As Single

read-write

Remarks

Image contrast is an important factor during both learning and inspection. The background and foreground information must be separated using an appropriate threshold, possibly determined automatically. After a threshold is given, the average gray level of the background and foreground areas are computed separately. These serve as reference gray levels to measure the image contrast and normalize the gray level quality indicators. The background and foreground reference gray levels are computed for both the template and sample images. They are used for contrast assessment as well as gray-level normalization.

EOCV.SampleContrast

Global contrast of the last inspected image, or **FLOAT32_UNDEFINED** if it has not been computed.

[VB6]

SampleContrast As Single

read-only

Remarks

The contrast is defined as the ratio of the reference gray-levels difference over their sum. This parameter is maximum (100 %) for a perfectly contrasted image (white on black).

EOCV.SampleForeground

Reference foreground gray levels determined during inspection of the sample.

[VB6]

SampleForeground As Single

read-write

Remarks

Image contrast is an important factor during both learning and inspection. The background and foreground information must be separated using an appropriate threshold, possibly determined automatically. After a threshold is given, the average gray level of the background and foreground areas are computed separately. These serve as reference gray levels to measure the image contrast and normalize the gray level quality indicators. The background and foreground reference gray levels are computed for both the template and sample images. They are used for contrast assessment as well as gray-level normalization.

EOCV.SampleThreshold

Threshold level applied to the sample during inspection.

[VB6]

SampleThreshold As EBW8

read-write

Remarks

Image contrast is an important factor during both learning and inspection. The background and foreground information must be separated using an appropriate threshold, possibly determined automatically. After a threshold is given, the average gray level of the background and foreground areas are computed separately. These serve as reference gray levels to measure the image contrast and normalize the gray level quality indicators. The background and foreground reference gray levels are computed for both the template and sample images. They are used for contrast assessment as well as gray-level normalization.

EOCV.Save

[VB6]

```
void Save(
    ESerializer serializer
)
void Save(
    String stream
)
```

Parameters

serializer

-

stream

-

EOCV.ScatterTextsCharsParameters

Copies the desired parameters from the temporary character to the characters specified by the selection mode.

[VB6]

```
void ScatterTextsCharsParameters(
    EOCVChar ocvChar,
    ESelectionFlag textsSelectionFlag,
    ESelectionFlag charsSelectionFlag
)
```

Parameters

ocvChar

Temporary [EOCVChar](#) object.

textsSelectionFlag

Selection mode for the texts, as defined by [ESelectionFlag](#).

charsSelectionFlag

Selection mode for the text characters, as defined by [ESelectionFlag](#).

Remarks

Only the parameters for which the value is not undefined are copied. Before setting parameters, be sure to call the [EOCChar::ResetParameters](#) member of the temporary [EOCChar](#) object. This will reset all parameters to the undefined value, thus avoiding unwanted copy.

EOCV.ScatterTextsParameters

Copies the desired parameters from the temporary text to the texts specified by the selection mode.

[VB6]

```
void ScatterTextsParameters(
    EOCVText Text,
    ESelectionFlag textsSelectionFlag
)
```

Parameters

Text

Selection mode for the texts, as defined by [ESelectionFlag](#).

textsSelectionFlag

Temporary [EOCText](#) object.

Remarks

Only the parameters for which the value is not undefined value are copied. Before setting parameters, be sure to call the [EOCText::ResetParameters](#) member of the temporary [EOCText](#) object. This will reset all parameters to the undefined value, thus avoiding unwanted copy.

EOCV.SelectSampleTexts

Toggles selection of the template texts whose bounding box intersects a given rectangle.

[VB6]

```
void SelectSampleTexts(
    Long originX,
    Long originY,
    Long width,
    Long height,
    ESelectionFlag textsSelectionFlag
)
```

Parameters

originX

Absolute coordinates of the selection rectangle (relative to the topmost parent image).

originY

Absolute coordinates of the selection rectangle (relative to the topmost parent image).

width

Limits of the selection rectangle.

height

Limits of the selection rectangle.

textsSelectionFlag

Selection mode for the template texts, as defined by [ESelectionFlag](#). By default, all texts are checked for intersection

Remarks

Unselected become selected and conversely. Reading or changing properties of the sample texts and the characters they contain can be done on basis of selection.

EOCV.SelectSampleTextsChars

Toggles selection of the template text characters whose bounding box intersects a given rectangle.

[VB6]

```
void SelectSampleTextsChars (
    Long originX,
    Long originY,
    Long width,
    Long height,
    ESelectionFlag textsSelectionFlag,
    ESelectionFlag charsSelectionFlag
)
```

Parameters

originX

Absolute coordinates of the selection rectangle (relative to the topmost parent image).

originY

Absolute coordinates of the selection rectangle (relative to the topmost parent image).

width

Limits of the selection rectangle.

height

Limits of the selection rectangle.

textsSelectionFlag

Selection mode for the sample texts, as defined by [ESelectionFlag](#). By default, all texts are checked for intersection

charsSelectionFlag

Selection mode for the sample text characters, as defined by [ESelectionFlag](#). By default, all characters of all texts are checked for intersection.

Remarks

Unselected become selected and conversely. Reading or changing properties of the sample texts and the characters they contain can be done on basis of selection.

EOCV.SelectTemplateChars

Toggles selection of the free characters whose bounding box intersects a given rectangle.

[VB6]

```
void SelectTemplateChars(
    Long originX,
    Long originY,
    Long width,
    Long height,
    ESelectionFlag charsSelectionFlag
)
```

Parameters

originX

Absolute coordinates of the selection rectangle (relative to the topmost parent image).

originY

Absolute coordinates of the selection rectangle (relative to the topmost parent image).

width

Limits of the selection rectangle.

height

Limits of the selection rectangle.

charsSelectionFlag

Selection mode for the free characters, as defined by [ESelectionFlag](#). By default, all free characters are processed.

Remarks

Unselected becomes selected and conversely. The free characters are sets of blobs taken from the free objects list. One step of the model definition is to specify which free objects will be handled by the OCV context and how they will be grouped to form characters. Grouping can be done automatically by using a overlapping criterion, or explicitly. When an object is added to a character, it is removed from the free objects list.

EOCV.SelectTemplateObjects

Toggles selection of the objects whose bounding box intersects a given rectangle.

[VB6]

```
void SelectTemplateObjects(
    Long originX,
    Long originY,
    Long width,
    Long height,
    ESelectionFlag objectsSelectionFlag
)
```

Parameters

originX

Absolute coordinates of the selection rectangle (relative to the topmost parent image).

originY

Absolute coordinates of the selection rectangle (relative to the topmost parent image).

width

Limits of the selection rectangle.

height

Limits of the selection rectangle.

objectsSelectionFlag

Selection mode for the objects, as defined by [ESelectionFlag](#). By default, all objects are processed.

Remarks

Unselected becomes selected and conversely. The free objects are blobs built using a separate coded image. One step of the model definition is to specify which objects will be handled by the OCV context.

EOCV.SelectTemplateTexts

Toggles selection of the template texts whose bounding box intersects a given rectangle.

[VB6]

```
void SelectTemplateTexts(
    Long originX,
    Long originY,
    Long width,
    Long height,
    ESelectionFlag textsSelectionFlag
)
```

Parameters

originX

Absolute coordinates of the selection rectangle (relative to the topmost parent image).

originY

Absolute coordinates of the selection rectangle (relative to the topmost parent image).

width

Limits of the selection rectangle.

height

Limits of the selection rectangle.

textsSelectionFlag

Selection mode for the template texts, as defined by [ESelectionFlag](#). By default, all texts are processed.

Remarks

Unselected becomes selected and conversely. The template texts are sets of characters taken from the free characters list. One step of the model definition is to specify which free characters will be handled by the OCV context and how they will be grouped to form texts. Grouping must be done explicitly. When a character is added to a text, it is removed from the free characters list.

EOCV.SetFreeChar

Sets an individual free character of the OCV context.

[VB6]

```
void SetFreeChar(
    Long index,
    EOCVChar freeChar
)
```

Parameters*index*

Index of the individual free character.

freeChar

New free character.

Remarks

The free characters are sets of blobs taken from the free objects list. One step of the model definition is to specify which free objects will be handled by the OCV context, and how they will be grouped to form characters. Grouping can be done automatically by using a overlapping criterion, or explicitly. When an object is added to a character, it is removed from the free objects list.

EOCV.SetText

Modifies the individual text at the given index by the given value.

[VB6]

```
void SetText(
    Long index,
    EOCVText text
)
```

Parameters*index*Index, between **0** and [EOCV::NumTexts](#) (excluded) of the individual text to be modified.*text*

The new text.

EOCV.SetTextCharParameters

Copies the desired parameters from the temporary character to the character of given index.

[VB6]

```
void SetTextCharParameters(
    EOCVChar ocvChar,
    Long textIndex,
    Long charIndex
)
```

Parameters

ocvChar

Temporary [EOCVChar](#) object.

textIndex

Character index in range **0** to **NumTexts-1**.

charIndex

Character index in range **0** to **NumTextChars-1**.

Remarks

Only the parameters for which the value is not undefined are copied. Before setting parameters, be sure to call the [EOCVChar::ResetParameters](#) member of the temporary [EOCVChar](#) object. This will reset all parameters to the undefined value, thus avoiding unwanted copy.

EOCV.SetTextParameters

Copies the desired parameters from the temporary text to the text of given index.

[VB6]

```
void SetTextParameters(
    EOCVText Text,
    Long textIndex
)
```

Parameters

Text

Text index in range **0** to **NumTexts**, excluded.

textIndex

Text index in range **0** to **NumTexts**, excluded.

Remarks

Only the parameters for which the value is not undefined value are copied. Before setting parameters, be sure to call the [EOCVText::ResetParameters](#) member of the temporary [EOCVText](#) object. This will reset all parameters to the undefined value, thus avoiding unwanted copy.

EOCV.StatisticsCount

Number of samples that have been taken into account in the statistics so far.

[VB6]

StatisticsCount As Long

read-only

Remarks

For averages to be computed, this member must amount to at least **1**. For standard deviations to be computed, this member must amount to at least **2**.

EOCV.TemplateBackground

Reference background gray levels determined during learning of the template.

[VB6]

TemplateBackground As Single

read-write

Remarks

Image contrast is an important factor during both learning and inspection. The background and foreground information must be separated using an appropriate threshold, possibly determined automatically. After a threshold is given, the average gray level of the background and foreground areas are computed separately. These serve as reference gray levels to measure the image contrast and normalize the gray level quality indicators. The background and foreground reference gray levels are computed for both the template and sample images. They are used for contrast assessment as well as gray-level normalization.

EOCV.TemplateContrast

Global contrast of the template image, or **FLOAT32_UNDEFINED** if it has not been computed.

[VB6]

TemplateContrast As Single

read-only

Remarks

The contrast is defined as the ratio of the reference gray-levels difference over their sum. This parameter is maximum (100 %) for a perfectly contrasted image (white on black).

EOCV.TemplateForeground

Reference foreground gray levels determined during learning of the template.

[VB6]**TemplateForeground As Single**

read-write

Remarks

Image contrast is an important factor during both learning and inspection. The background and foreground information must be separated using an appropriate threshold, possibly determined automatically. After a threshold is given, the average gray level of the background and foreground areas are computed separately. These serve as reference gray levels to measure the image contrast and normalize the gray level quality indicators. The background and foreground reference gray levels are computed for both the template and sample images. They are used for contrast assessment as well as gray-level normalization.

EOCV.TemplateImage

Source template image associated to the OCV context during model edition.

[VB6]**TemplateImage As EROIBW8**

read-write

Remarks

This property must be set as a first step before any model edition operation. During edition, the source image should not be altered. Before learning, the OCV context must have access to the template image to pre-process it. After learning, the OCV context keeps an internal copy for comparison purposes.

EOCV.TemplateThreshold

Threshold level applied to the template during learning.

[VB6]

TemplateThreshold As EBW8

read-write

Remarks

Image contrast is an important factor during both learning and inspection. The background and foreground information must be separated using an appropriate threshold, possibly determined automatically. After a threshold is given, the average gray level of the background and foreground areas are computed separately. These serve as reference gray levels to measure the image contrast and normalize the gray level quality indicators. The background and foreground reference gray levels are computed for both the template and sample images. They are used for contrast assessment as well as gray-level normalization.

EOCV.UpdateStatistics

Adds the parameters of the last inspection to the statistics.

[VB6]

```
void UpdateStatistics()  
{}
```

Remarks

This function must be called explicitly for the current sample to be taken into account.

EOCV.UsedQualityIndicators

Choice of quality indicators.

[VB6]

UsedQualityIndicators As Long

read-write

Remarks

For efficiency reasons, the quality indicators can be computed on demand only. This property is the logical combination (bitwise OR) of the values in [EQualityIndicator](#). When a bit is set in this mask, the corresponding feature is computed.

EOCV.WhiteOnBlack

Flag indicating whether the learning and inspection steps use white characters on a black background, or black characters on a white background.

[VB6]

WhiteOnBlack As Boolean

read-write

Remarks

If **TRUE** (default), the learning and inspection steps use white characters on a black background. If **FALSE**, black characters on a white background are taken into account. It is important to note that this parameter affects the learning *and* the inspection steps. Moreover, the use of the "black characters on a white background" mode requires the choice of the correct [ECodedImage](#) class.

EOCVChar Class

Holds all information related to a single character, such as nominal and average position, reference quality indicators,... During the learning phases, it also keeps the list of its constituent blobs.

Remarks

The statistics available for each character are the average, standard deviation (unbiased), minimum and maximum computed on the corresponding parameters for all images for which [EOCV::UpdateStatistics](#) has been invoked after inspection. The average is only available when at least one set of results has been accumulated. The standard deviation is only available when at least two sets of results have been accumulated.

PROPERTIES

BackgroundAreaAverage	Background area average.
BackgroundAreaDeviation	Background area standard deviation.
BackgroundAreaTolerance	Maximum allowed difference between the sample and template background areas.
BackgroundSumAverage	Background sum average.
BackgroundSumDeviation	Background sum standard deviation.
BackgroundSumTolerance	Maximum allowed difference between the sample and template background sums.
Correlation	Normalized correlation involving the template and sample character images.
CorrelationAverage	Correlation average.
CorrelationDeviation	Correlation standard deviation.

CorrelationTolerance	Maximum allowed difference between the normalized correlation and unity.
Diagnostics	Logical combination (bitwise OR) of defects, as defined by EDiagnostic .
ForegroundAreaAverage	Foreground area average.
ForegroundAreaDeviation	Foreground area standard deviation.
ForegroundAreaTolerance	Maximum allowed difference between the sample and template foreground areas.
ForegroundSumAverage	Foreground sum average.
ForegroundSumDeviation	Foreground sum standard deviation.
ForegroundSumTolerance	Maximum allowed difference between the sample and template foreground sums.
LocationScoreAverage	Location score average.
LocationScoreDeviation	Location score standard deviation.
LocationScoreTolerance	Allowed absolute difference between the template and sample scores.

MarginWidth	Width of the extra space to be used around the characters bounding box (on all four sides) when computing a quality indicator.
NumContourPoints	Number of contour points of this character used for location.
SampleBackgroundArea	Number of background pixels of this character corresponding to a background pixel of the template character.
SampleBackgroundSum	Sum of the normalized gray-level values of the pixels of this character corresponding to a background pixel of the template character.
SampleForegroundArea	Number of foreground pixels of this character corresponding to a foreground pixel of the template character.
SampleForegroundSum	Sum of the normalized gray-level values of the pixels of this character corresponding to a foreground pixel of the template character.
SampleLocationScore	Value of the location score measured on the sample during inspection.
Selected	Selection state: TRUE when selected.

ShiftX	Measured horizontal translation.
ShiftXAverage	Shift X average.
ShiftXBias	Horizontal translation bias.
ShiftXDeviation	Shift X standard deviation.
ShiftXMax	Maximum Shift X.
ShiftXMin	Minimum Shift X.
ShiftXStride	First pass stride for the horizontal translation.
ShiftXTolerance	Horizontal translation tolerance.
ShiftY	Measured vertical translation.
ShiftYAverage	Shift Y average.
ShiftYBias	Vertical translation bias. 0 corresponds to the nominal position.
ShiftYDeviation	Shift Y standard deviation.

ShiftYMax	Maximum Shift Y.
ShiftYMin	Minimum Shift Y.
ShiftYStride	First pass stride for the vertical translation.
ShiftYTolerance	Vertical translation tolerance.
TemplateBackgroundArea	Number of pixels in the background of this character.
TemplateBackgroundSum	Sum of the normalized gray-level values of the background pixels of this character.
TemplateForegroundArea	Number of pixels in the template foreground of this character.
TemplateForegroundSum	Sum of the normalized gray-level values of the foreground pixels of this character.
TemplateLocationScore	Value of the location score measured on the template during learning.
WhiteOnBlack	Character contrast: TRUE for light characters on a dark background.

METHODS**EOCVChar**

Constructs an OCVChar context.

operator=

Copies all the data from another EOCVChar object into the current EOCVChar object

ResetParameters

Sets all character parameters to the undefined value for use before a [EOCV::SetTextCharParameters](#) or [EOCV::ScatterTextsCharsParameters](#) operation.

CVChar.BackgroundAreaAverage

Background area average.

[VB6]

BackgroundAreaAverage As Single

read-write

EOCVChar.BackgroundAreaDeviation

Background area standard deviation.

[VB6]

BackgroundAreaDeviation As Single

read-write

EOCVChar.BackgroundAreaTolerance

Maximum allowed difference between the sample and template background areas.

[VB6]

BackgroundAreaTolerance As Long

read-write

EOCVChar.BackgroundSumAverage

Background sum average.

[VB6]

BackgroundSumAverage As Single

read-write

EOCVChar.BackgroundSumDeviation

Background sum standard deviation.

[VB6]

BackgroundSumDeviation As Single

read-write

EOCVChar.BackgroundSumTolerance

Maximum allowed difference between the sample and template background sums.

[VB6]

BackgroundSumTolerance As Single

read-write

EOCVChar.Correlation

Normalized correlation involving the template and sample character images.

[VB6]

Correlation As Single

read-write

EOCVChar.CorrelationAverage

Correlation average.

[VB6]

CorrelationAverage As Single

read-write

EOCVChar.CorrelationDeviation

Correlation standard deviation.

[VB6]

CorrelationDeviation As Single

read-write

EOCVChar.CorrelationTolerance

Maximum allowed difference between the normalized correlation and unity.

[VB6]

CorrelationTolerance As Single

read-write

EOCVChar.Diagnostics

Logical combination (bitwise OR) of defects, as defined by [EDiagnostic](#).

[VB6]

Diagnostics As Long

read-write

Remarks

After inspection, each character is tagged with a logical combination of diagnostics, each corresponding to a kind of defect.

EOCVChar.EOCVChar

Constructs an OCVChar context.

```
[VB6]  
void EOCVChar()  
)  
  
void EOCVChar(  
EOCVChar other  
)
```

Parameters

other

EOCVChar object to be copied.

Remarks

Default and copy constructors.

EOCVChar.ForegroundAreaAverage

Foreground area average.

```
[VB6]  
ForegroundAreaAverage As Single
```

read-write

EOCVChar.ForegroundAreaDeviation

Foreground area standard deviation.

[VB6]

ForegroundAreaDeviation As Single

read-write

EOCVChar.ForegroundAreaTolerance

Maximum allowed difference between the sample and template foreground areas.

[VB6]

ForegroundAreaTolerance As Long

read-write

EOCVChar.ForegroundSumAverage

Foreground sum average.

[VB6]

ForegroundSumAverage As Single

read-write

EOCVChar.ForegroundSumDeviation

Foreground sum standard deviation.

[VB6]

ForegroundSumDeviation As Single

read-write

EOCVChar.ForegroundSumTolerance

Maximum allowed difference between the sample and template foreground sums.

[VB6]

ForegroundSumTolerance As Single

read-write

EOCVChar.LocationScoreAverage

Location score average.

[VB6]

LocationScoreAverage As Single

read-write

EOCVChar.LocationScoreDeviation

Location score standard deviation.

[VB6]

LocationScoreDeviation As Single

read-write

EOCVChar.LocationScoreTolerance

Allowed absolute difference between the template and sample scores.

[VB6]

LocationScoreTolerance As Single

read-write

Remarks

When the sample value lies outside the acceptance interval, a character not found diagnostic is issued. During the location phase, a score is computed on the sample image. During learning, the same score is measured on the template image to serve as a reference. The closer the template and sample scores, the more successful the location. The location score is a first indication on the conformance of the inspected sample. In particular, a very small sample score may indicate that the character is absent.

EOCVChar.MarginWidth

Width of the extra space to be used around the characters bounding box (on all four sides) when computing a quality indicator.

[VB6]

MarginWidth As Long

read-write

EOCVChar.NumContourPoints

Number of contour points of this character used for location.

[VB6]

NumContourPoints As Long

read-write

Remarks

The location process relies on points from the external contours of the character constituent blobs. The number of points to be used can be adjusted. The smaller this value, the faster the location, but a too small value can cause mismatches.

EOCVChar.operator=

Copies all the data from another EOCVChar object into the current EOCVChar object

[VB6]

```
EOCVChar operator=(
    EOCVChar other
)
```

Parameters

other

EOCVChar object to be copied

EOCVChar.ResetParameters

Sets all character parameters to the undefined value for use before a [EOCV::SetTextCharParameters](#) or [EOCV::ScatterTextsCharsParameters](#) operation.

[VB6]

```
void ResetParameters(
)
```

EOCVChar.SampleBackgroundArea

Number of background pixels of this character corresponding to a background pixel of the template character.

[VB6]

```
SampleBackgroundArea As Long  
read-write
```

EOCVChar.SampleBackgroundSum

Sum of the normalized gray-level values of the pixels of this character corresponding to a background pixel of the template character.

[VB6]

SampleBackgroundSum As Single

read-write

EOCVChar.SampleForegroundArea

Number of foreground pixels of this character corresponding to a foreground pixel of the template character.

[VB6]

SampleForegroundArea As Long

read-write

EOCVChar.SampleForegroundSum

Sum of the normalized gray-level values of the pixels of this character corresponding to a foreground pixel of the template character.

[VB6]

SampleForegroundSum As Single

read-write

EOCVChar.SampleLocationScore

Value of the location score measured on the sample during inspection.

[VB6]

SampleLocationScore As Single

read-write

Remarks

During the location phase, a score is computed on the sample image. During learning, the same score is measured on the template image to serve as a reference. The closer the template and sample scores, the more successful the location. The location score is a first indication on the conformance of the inspected sample. In particular, a very small sample score may indicate that the character is absent.

EOCVChar.Selected

Selection state: **TRUE** when selected.

[VB6]

Selected As Boolean

read-write

EOCVChar.ShiftX

Measured horizontal translation.

[VB6]

ShiftX As Single

read-write

EOCVChar.ShiftXAverage

Shift X average.

[VB6]

ShiftXAverage As Single

read-write

EOCVChar.ShiftXBias

Horizontal translation bias.

[VB6]

ShiftXBias As Single

read-write

Remarks

0 corresponds to the nominal position.

EOCVChar.ShiftXDeviation

Shift X standard deviation.

[VB6]

ShiftXDeviation As Single

read-write

EOCVChar.ShiftXMax

Maximum Shift X.

[VB6]

ShiftXMax As Single

read-write

EOCVChar.ShiftXMin

Minimum Shift X.

[VB6]

ShiftXMin As Single

read-write

EOCVChar.ShiftXStride

First pass stride for the horizontal translation.

[VB6]

ShiftXStride As Long

read-write

EOCVChar.ShiftXTolerance

Horizontal translation tolerance.

[VB6]

ShiftXTolerance As Single

read-write

EOCVChar.ShiftY

Measured vertical translation.

[VB6]

ShiftY As Single

read-write

EOCVChar.ShiftYAverage

Shift Y average.

[VB6]

ShiftYAverage As Single

read-write

EOCVChar.ShiftYBias

Vertical translation bias. **0** corresponds to the nominal position.

[VB6]

ShiftYBias As Single

read-write

EOCVChar.ShiftYDeviation

Shift Y standard deviation.

[VB6]

ShiftYDeviation As Single

read-write

EOCVChar.ShiftYMax

Maximum Shift Y.

[VB6]

ShiftYMax As Single

read-write

EOCVChar.ShiftYMin

Minimum Shift Y.

[VB6]

ShiftYMin As Single

read-write

EOCVChar.ShiftYStride

First pass stride for the vertical translation.

[VB6]

ShiftYStride As Long

read-write

EOCVChar.ShiftYTolerance

Vertical translation tolerance.

[VB6]

ShiftYTolerance As Single

read-write

EOCVChar.TemplateBackgroundArea

Number of pixels in the background of this character.

[VB6]

TemplateBackgroundArea As Long

read-write

EOCVChar.TemplateBackgroundSum

Sum of the normalized gray-level values of the background pixels of this character.

[VB6]

TemplateBackgroundSum As Single

read-write

EOCVChar.TemplateForegroundArea

Number of pixels in the template foreground of this character.

[VB6]

TemplateForegroundArea As Long

read-write

EOCVChar.TemplateForegroundSum

Sum of the normalized gray-level values of the foreground pixels of this character.

[VB6]

TemplateForegroundSum As Single

read-write

EOCVChar.TemplateLocationScore

Value of the location score measured on the template during learning.

[VB6]

TemplateLocationScore As Single

read-write

Remarks

If necessary, this value may be set explicitly. During the location phase, a score is computed on the sample image. During learning, the same score is measured on the template image to serve as a reference. The closer the template and sample scores, the more successful the location. The location score is a first indication on the conformance of the inspected sample. In particular, a very small sample score may indicate that the character is absent.

EOCVChar.WhiteOnBlack

Character contrast: **TRUE** for light characters on a dark background.

[VB6]

WhiteOnBlack As Boolean

read-write

EOCVText Class

Holds all information related to a single piece of text, such as nominal and average position, reference quality indicators,... It also keeps the list of its constituent characters.

Remarks

Each text can be translated horizontally and vertically, rotated, scaled horizontally and vertically and sheared with respect to its nominal position, to cope with mechanical displacement of the marking device. Location is performed in the range **Bias +/- Tolerance** around the nominal position. To speed up location, a two pass search may be performed, first with a large stride, then with a unit stride.

Note. The stride parameters apply to the two translation degrees of freedom only. A set of parameters are computed on demand (see Inspection Options) during inspection. The values of these parameters are used to detect defects of various kinds by checking that they remain in a given tolerance interval. If not, a diagnostic is issued. Note that the quality indicators associated with the texts are the sums of the corresponding parameters for each of the constituent characters. The statistics available for each text are the average, standard deviation (unbiased), minimum and maximum computed on the corresponding parameters for all images for which [EOCV::UpdateStatistics](#) has been invoked after inspection. The average is only available when at least one set of results has been accumulated. The standard deviation is only available when at least two sets of results have been accumulated.

PROPERTIES

BackgroundAreaAverage

Background area average.

BackgroundAreaDeviation	Background area standard deviation.
BackgroundAreaTolerance	Maximum allowed difference between the sample and template background areas.
BackgroundSumAverage	Background sum average.
BackgroundSumDeviation	Background sum standard deviation.
BackgroundSumTolerance	Maximum allowed difference between the sample and template background sums.
Correlation	Normalized correlation involving the template and sample images of the characters of this text.
CorrelationAverage	Correlation average.
CorrelationDeviation	Correlation standard deviation.
CorrelationTolerance	Maximum allowed difference between the normalized correlation and unity.
Diagnostics	Logical combination (bitwise OR) of defects, as defined by EDiagnostic .
ForegroundAreaAverage	Foreground area average.

ForegroundAreaDeviation	Foreground area standard deviation.
ForegroundAreaTolerance	Maximum allowed difference between the sample and template foreground areas.
ForegroundSumAverage	Foreground sum average.
ForegroundSumDeviation	Foreground sum standard deviation.
ForegroundSumTolerance	Maximum allowed difference between the sample and template foreground sums.
IsotropicScaling	Flag indicating whether the scaling degree of freedom should be considered isotropic (ScaleX and ScaleY identical) or not.
LocationScoreAverage	Location score average.
LocationScoreDeviation	Location score standard deviation.
LocationScoreTolerance	Allowed absolute difference between the template and sample scores.
MarginWidth	Unused.
NumContourPoints	Number of contour points used for location of this text.

SampleBackgroundArea	Number of background pixels of this text corresponding to a background pixel of the characters of the template text.
SampleBackgroundSum	Sum of the normalized gray-level values of the pixels of this text corresponding to a background pixel of the characters of the template text.
SampleForegroundArea	Number of foreground pixels of this text corresponding to a foreground pixel of the characters of the template text.
SampleForegroundSum	Sum of the normalized gray-level values of the pixels of this text corresponding to a foreground pixel of the characters of the template text.
SampleLocationScore	Value of the location score measured on the sample during inspection.
ScaleX	Measured horizontal scaling, expressed as a dimensionless ratio.
ScaleXAverage	Scale X average.
ScaleXBias	Horizontal scaling bias.
ScaleXCount	Number of values to be tried in the Bias +/- Tolerance range, bounds included.

ScaleXDeviation	Scale X standard deviation.
ScaleXMax	Maximum Y-scale.
ScaleXMin	Minimum X-scale.
ScaleXTolerance	Horizontal scaling tolerance.
ScaleY	Measured vertical scaling, expressed as a dimensionless ratio.
ScaleYAverage	Scale Y average.
ScaleYBias	Vertical scaling bias.
ScaleYCount	Number of values to be tried in the Bias +/- Tolerance range, bounds included.
ScaleYDeviation	Scale Y standard deviation.
ScaleYMax	Maximum Y-scale.
ScaleYMin	Minimum Y-scale.
ScaleYTolerance	Vertical scaling tolerance.

Selected	Selection state. TRUE when selected.
Shear	Measured shearing, clockwise from the vertical direction, expressed in the current angle unit.
ShearAverage	Shear average.
ShearBias	Shearing bias.
ShearCount	Number of values to be tried in the Bias +/- Tolerance range, bounds included.
ShearDeviation	Shear standard deviation.
ShearMax	Maximum shear.
ShearMin	Minimum shear.
ShearTolerance	Shearing tolerance.
ShiftX	Measured horizontal translation, in pixels.
ShiftXAverage	Shift X average.
ShiftXBias	Horizontal translation bias.

ShiftXDeviation	Shift X standard deviation.
ShiftXMax	Maximum Shift Y.
ShiftXMin	Minimum Shift X.
ShiftXStride	First pass stride for the horizontal translation.
ShiftXTolerance	Horizontal translation tolerance.
ShiftY	Measured vertical translation, in pixels.
ShiftYAverage	Shift Y average.
ShiftYBias	Vertical translation bias.
ShiftYDeviation	Shift Y standard deviation.
ShiftYMax	Maximum Shift Y.
ShiftYMin	Minimum Shift Y.
ShiftYStride	First pass stride for the vertical translation.

ShiftYTolerance	Vertical translation tolerance.
Skew	Measured rotation, clockwise from the horizontal direction, expressed in the current angle unit.
SkewAverage	Skew average.
SkewBias	Rotation bias.
SkewCount	Number of values to be tried in the Bias +/- Tolerance range, bounds included.
SkewDeviation	Skew standard deviation.
SkewMax	Maximum skew.
SkewMin	Minimum skew.
SkewTolerance	Rotation tolerance.
TemplateBackgroundArea	Number of pixels in the background of the characters of this text.
TemplateBackgroundSum	Sum of the normalized gray-level values of the background pixels of the characters of this text.

TemplateForegroundArea	Number of pixels in the foreground of all characters of this text.
TemplateForegroundSum	Sum of the normalized gray-level values of the foreground pixels of the characters of this text.
TemplateLocationScore	M Value of the location score measured on the template during learning. If necessary, this value E may be set explicitly.
THODS	
EOCVText	Constructs an OCVText context.
operator=	Copies all the data from another EOCVText object into the current EOCVText object
ResetParameters	E Sets all text parameters to the undefined value for use before a EOCV::SetTextParameters or EOCV::ScatterTextsParameters operation. O

CVText.BackgroundAreaAverage

Background area average.

[VB6]

```
BackgroundAreaAverage As Single
```

read-write

EOCVText.BackgroundAreaDeviation

Background area standard deviation.

[VB6]

BackgroundAreaDeviation As Single

read-write

EOCVText.BackgroundAreaTolerance

Maximum allowed difference between the sample and template background areas.

[VB6]

BackgroundAreaTolerance As Long

read-write

EOCVText.BackgroundSumAverage

Background sum average.

[VB6]

BackgroundSumAverage As Single

read-write

EOCVText.BackgroundSumDeviation

Background sum standard deviation.

[VB6]

BackgroundSumDeviation As Single

read-write

EOCVText.BackgroundSumTolerance

Maximum allowed difference between the sample and template background sums.

[VB6]

BackgroundSumTolerance As Single

read-write

EOCVText.Correlation

Normalized correlation involving the template and sample images of the characters of this text.

[VB6]

Correlation As Single

read-write

EOCVText.CorrelationAverage

Correlation average.

[VB6]

CorrelationAverage As Single

read-write

EOCVText.CorrelationDeviation

Correlation standard deviation.

[VB6]

CorrelationDeviation As Single

read-write

EOCVText.CorrelationTolerance

Maximum allowed difference between the normalized correlation and unity.

[VB6]

CorrelationTolerance As Single

read-write

EOCVText.Diagnostics

Logical combination (bitwise OR) of defects, as defined by [EDiagnostic](#).

[VB6]

Diagnostics As Long

read-write

Remarks

After inspection, each text is tagged with a logical combination of diagnostics, each corresponding to a kind of defect. The defects of the characters belonging to this texts are also added to this combination.

EOCVText.EOCVText

Constructs an OCVText context.

[VB6]

```
void EOCVText()
}

void EOCVText(
    EOCVText ocvText
)
```

Parameters

ocvText

EOCVText object to be copied.

Remarks

Default and copy constructors.

EOCVText.ForegroundAreaAverage

Foreground area average.

[VB6]

ForegroundAreaAverage As Single

read-write

EOCVText.ForegroundAreaDeviation

Foreground area standard deviation.

[VB6]

ForegroundAreaDeviation As Single

read-write

EOCVText.ForegroundAreaTolerance

Maximum allowed difference between the sample and template foreground areas.

[VB6]

ForegroundAreaTolerance As Long

read-write

EOCVText.ForegroundSumAverage

Foreground sum average.

[VB6]

ForegroundSumAverage As Single

read-write

EOCVText.ForegroundSumDeviation

Foreground sum standard deviation.

[VB6]

ForegroundSumDeviation As Single

read-write

EOCVText.ForegroundSumTolerance

Maximum allowed difference between the sample and template foreground sums.

[VB6]

ForegroundSumTolerance As Single

read-write

EOCVText.IsotropicScaling

Flag indicating whether the scaling degree of freedom should be considered isotropic (**ScaleX** and **ScaleY** identical) or not.

[VB6]

IsotropicScaling As Boolean

read-write

Remarks

In most cases, isotropic scaling (default mode), is recommended. Isotropic scaling executes faster and is more realistic. In case of isotropic scaling search, the **ScaleY...** values are meaningless.

EOCVText.LocationScoreAverage

Location score average.

[VB6]

LocationScoreAverage As Single

read-write

EOCVText.LocationScoreDeviation

Location score standard deviation.

[VB6]

LocationScoreDeviation As Single

read-write

EOCVText.LocationScoreTolerance

Allowed absolute difference between the template and sample scores.

[VB6]

LocationScoreTolerance As Single

read-write

Remarks

When the sample value lies outside the acceptance interval, a text not found diagnostic is issued.

EOCVText.MarginWidth

Unused.

[VB6]

MarginWidth As Long

read-write

EOCVText.NumContourPoints

Number of contour points used for location of this text.

[VB6]

NumContourPoints As Long

read-write

EOCVText.operator=

Copies all the data from another EOCVText object into the current EOCVText object

[VB6]

```
EOCVText operator=(  
    EOCVText other  
)
```

Parameters

other

EOCVText object to be copied

EOCVText.ResetParameters

Sets all text parameters to the undefined value for use before a [EOCV::SetTextParameters](#) or [EOCV::ScatterTextsParameters](#) operation.

[VB6]

```
void ResetParameters(  
)
```

EOCVText.SampleBackgroundArea

Number of background pixels of this text corresponding to a background pixel of the characters of the template text.

[VB6]

SampleBackgroundArea As Long

read-write

EOCVText.SampleBackgroundSum

Sum of the normalized gray-level values of the pixels of this text corresponding to a background pixel of the characters of the template text.

[VB6]

SampleBackgroundSum As Single

read-write

EOCVText.SampleForegroundArea

Number of foreground pixels of this text corresponding to a foreground pixel of the characters of the template text.

[VB6]

SampleForegroundArea As Long

read-write

EOCVText.SampleForegroundSum

Sum of the normalized gray-level values of the pixels of this text corresponding to a foreground pixel of the characters of the template text.

[VB6]

SampleForegroundSum As Single

read-write

EOCVText.SampleLocationScore

Value of the location score measured on the sample during inspection.

[VB6]

SampleLocationScore As Single

read-write

EOCVText.ScaleX

Measured horizontal scaling, expressed as a dimensionless ratio.

[VB6]

ScaleX As Single

read-write

EOCVText.ScaleXAverage

Scale X average.

[VB6]

ScaleXAverage As Single

read-write

EOCVText.ScaleXBias

Horizontal scaling bias.

[VB6]

ScaleXBias As Single

read-write

Remarks

0 corresponds to the nominal, true scale factor.

EOCVText.ScaleXCount

Number of values to be tried in the Bias +/- Tolerance range, bounds included.

[VB6]

ScaleXCount As Long

read-write

EOCVText.ScaleXDeviation

Scale X standard deviation.

[VB6]

ScaleXDeviation As Single

read-write

EOCVText.ScaleXMax

Maximum Y-scale.

[VB6]

ScaleXMax As Single

read-write

EOCVText.ScaleXMin

Minimum X-scale.

[VB6]

ScaleXMin As Single

read-write

EOCVText.ScaleXTolerance

Horizontal scaling tolerance.

[VB6]

ScaleXTolerance As Single

read-write

EOCVText.ScaleY

Measured vertical scaling, expressed as a dimensionless ratio.

[VB6]

ScaleY As Single

read-write

EOCVText.ScaleYAverage

Scale Y average.

[VB6]

ScaleYAverage As Single

read-write

EOCVText.ScaleYBias

Vertical scaling bias.

[VB6]

ScaleYBias As Single

read-write

Remarks

0 corresponds to the nominal, true scale factor.

EOCVText.ScaleYCount

Number of values to be tried in the Bias +/- Tolerance range, bounds included.

[VB6]

ScaleYCount As Long

read-write

EOCVText.ScaleYDeviation

Scale Y standard deviation.

[VB6]

ScaleYDeviation As Single

read-write

EOCVText.ScaleYMax

Maximum Y-scale.

[VB6]

ScaleYMax As Single

read-write

EOCVText.ScaleYMin

Minimum Y-scale.

[VB6]

ScaleYMin As Single

read-write

EOCVText.ScaleYTolerance

Vertical scaling tolerance.

[VB6]

ScaleYTolerance As Single

read-write

EOCVText.Selected

Selection state. **TRUE** when selected.

[VB6]

Selected As Boolean

read-write

EOCVText.Shear

Measured shearing, clockwise from the vertical direction, expressed in the current angle unit.

[VB6]

Shear As Single

read-write

EOCVText.ShearAverage

Shear average.

[VB6]

ShearAverage As Single

read-write

EOCVText.ShearBias

Shearing bias.

[VB6]

ShearBias As Single

read-write

Remarks

0 corresponds to the nominal, upright position.

EOCVText.ShearCount

Number of values to be tried in the Bias +/- Tolerance range, bounds included.

[VB6]

ShearCount As Long

read-write

EOCVText.ShearDeviation

Shear standard deviation.

[VB6]

ShearDeviation As Single

read-write

EOCVText.ShearMax

Maximum shear.

[VB6]

ShearMax As Single

read-write

EOCVText.ShearMin

Minimum shear.

[VB6]

ShearMin As Single

read-write

EOCVText.ShearTolerance

Shearing tolerance.

[VB6]

ShearTolerance As Single

read-write

EOCVText.ShiftX

Measured horizontal translation, in pixels.

[VB6]

ShiftX As Single

read-write

EOCVText.ShiftXAverage

Shift X average.

[VB6]

ShiftXAverage As Single

read-write

EOCVText.ShiftXBias

Horizontal translation bias.

[VB6]

ShiftXBias As Single

read-write

Remarks

0 corresponds to the nominal position.

EOCVText.ShiftXDeviation

Shift X standard deviation.

[VB6]

ShiftXDeviation As Single

read-write

EOCVText.ShiftXMax

Maximum Shift Y.

[VB6]

ShiftXMax As Single

read-write

EOCVText.ShiftXMin

Minimum Shift X.

[VB6]

ShiftXMin As Single

read-write

EOCVText.ShiftXStride

First pass stride for the horizontal translation.

[VB6]

ShiftXStride As Long

read-write

EOCVText.ShiftXTolerance

Horizontal translation tolerance.

[VB6]

ShiftXTolerance As Single

read-write

EOCVText.ShiftY

Measured vertical translation, in pixels.

[VB6]

ShiftY As Single

read-write

EOCVText.ShiftYAverage

Shift Y average.

[VB6]

ShiftYAverage As Single

read-write

EOCVText.ShiftYBias

Vertical translation bias.

[VB6]

ShiftYBias As Single

read-write

Remarks

0 corresponds to the nominal position.

EOCVText.ShiftYDeviation

Shift Y standard deviation.

[VB6]

ShiftYDeviation As Single

read-write

EOCVText.ShiftYMax

Maximum Shift Y.

[VB6]

ShiftYMax As Single

read-write

EOCVText.ShiftYMin

Minimum Shift Y.

[VB6]

ShiftYMin As Single

read-write

EOCVText.ShiftYStride

First pass stride for the vertical translation.

[VB6]

ShiftYStride As Long

read-write

EOCVText.ShiftYTolerance

Vertical translation tolerance.

[VB6]

ShiftYTolerance As Single

read-write

EOCVText.Skew

Measured rotation, clockwise from the horizontal direction, expressed in the current angle unit.

[VB6]

Skew As Single

read-write

EOCVText.SkewAverage

Skew average.

[VB6]

SkewAverage As Single

read-write

EOCVText.SkewBias

Rotation bias.

[VB6]

SkewBias As Single

read-write

Remarks

0 corresponds to the nominal, horizontal position.

EOCVText.SkewCount

Number of values to be tried in the Bias +/- Tolerance range, bounds included.

[VB6]

SkewCount As Long

read-write

EOCVText.SkewDeviation

Skew standard deviation.

[VB6]

SkewDeviation As Single

read-write

EOCVText.SkewMax

Maximum skew.

[VB6]

SkewMax As Single

read-write

EOCVText.SkewMin

Minimum skew.

[VB6]

SkewMin As Single

read-write

EOCVText.SkewTolerance

Rotation tolerance.

[VB6]

SkewTolerance As Single

read-write

EOCVText.TemplateBackgroundArea

Number of pixels in the background of the characters of this text.

[VB6]

TemplateBackgroundArea As Long

read-write

EOCVText.TemplateBackgroundSum

Sum of the normalized gray-level values of the background pixels of the characters of this text.

[VB6]

TemplateBackgroundSum As Single

read-write

EOCVText.TemplateForegroundArea

Number of pixels in the foreground of all characters of this text.

[VB6]

TemplateForegroundArea As Long

read-write

EOCVText.TemplateForegroundSum

Sum of the normalized gray-level values of the foreground pixels of the characters of this text.

[VB6]

TemplateForegroundSum As Single

read-write

EOCVText.TemplateLocationScore

Value of the location score measured on the template during learning. If necessary, this value may be set explicitly.

[VB6]

TemplateLocationScore As Single

read-write

EPathVector Class

Vector objects are used to store 1-dimensional data.

Remarks

Using vectors is very similar to using 1-dimensional arrays, except that the size can vary at runtime. Memory allocation is handled internally. * To create a vector, use its constructor. * To fill a vector with values, first empty it, using the [EVector::Empty](#) member, and then add elements one at time at the tail by calling the [EPathVector::AddElement](#) member. * To access a vector element, either for reading or writing, use the [EPathVector.operator\[\]@](#). * To inquire for the current number of elements, use member [EVector::NumElements](#).

Base Class: EVector

PROPERTIES

Closed

-

RawDataPtr

M

Pointer to the vector data.

E

THODS

AddElement

Appends (adds at the tail) an element to the vector.

Draw

Draws a plot of the vector element values.

DrawWithCurrentPen

Draws a plot of the vector element values.

EPathVector

Constructs a vector.

GetElement

Returns the vector element at the given index.

operator[]

Gives access to the vector element at the given index.

operator=

Copies all the data from another EPathVector object into the current EPathVector object

SetElement

E Modifies the vector element at the given index by the given value.

P

EPathVector.AddElement

Appends (adds at the tail) an element to the vector.

[VB6]

```
void AddElement(  
    EPath element  
)
```

Parameters

element

The element to be added.

EPathVector.Closed

-

[VB6]

Closed As Boolean

read-write

EPathVector.Draw

Draws a plot of the vector element values.

[VB6]

```

void Draw(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

void Draw(
    Long graphicContext,
    ERGBColor color,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

void Draw(
    EDrawAdapter graphicContext,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)

```

Parameters

graphicContext

Handle of the device context on which to draw.

zoomX

Zooming factor along the X axis (**1.0f** means no zoom).

zoomY

Zooming factor along the Y axis (**1.0f** means no zoom).

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

color

The color in which to draw the overlay.

Remarks

The vector draws line segment between the element coordinates. The drawing appears on the image itself.

EPathVector.DrawWithCurrentPen

Draws a plot of the vector element values.

[VB6]

```
void DrawWithCurrentPen(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

zoomX

Zooming factor along the X axis (**1.0f** means no zoom).

zoomY

Zooming factor along the Y axis (**1.0f** means no zoom).

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

Remarks

The vector draws line segment between the element coordinates. The drawing appears on the image itself.

EPathVector.EPathVector

Constructs a vector.

```
[VB6]
void EPathVector(
)
void EPathVector(
    Long maxNumberOfElements
)
void EPathVector(
    EPathVector other
)
```

Parameters

maxNumberOfElements

Optionally, memory can be pre-allocated to accommodate a given number of elements.

other

EPathVector object to be copied

EPathVector.GetElement

Returns the vector element at the given index.

```
[VB6]
EPath GetElement(
    Long index
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded) of the element to be accessed.

Remarks

If the given index is outside the bounds of the vector, the error code [EError_Parameter1OutOfRange](#) is set.

EPathVector.operator[]

Gives access to the vector element at the given index.

[VB6]

```
EPath operator[](  
    Long index  
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded) of the element to be accessed.

EPathVector.operator=

Copies all the data from another EPathVector object into the current EPathVector object

[VB6]

```
EPathVector operator=(  
    EPathVector other  
)
```

Parameters

other

EPathVector object to be copied

EPathVector.RawDataPtr

Pointer to the vector data.

[VB6]

RawDataPtr As Long

read-only

EPathVector.SetElement

Modifies the vector element at the given index by the given value.

[VB6]

```
void SetElement(  
    Long index,  
    EPath value  
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded), of the element to be modified.

value

The new value for the element.

Remarks

If the given index is outside the bounds of the vector, the error code [EError_Parameter1OutOfRange](#) is set.

EPatternFinder Class

Manages a complete finding context in EasyFind.

Base Class: [EPointShape](#)

PROPERTIES

[Angle](#)

-

[AngleBias](#)

Angle bias, expressed in the current angle unit.

[AngleSearchExtent](#)

The angular extension of the search neighborhood.
See the [EPatternFinder::LocalSearchMode](#) property
description for further details.

[AngleTolerance](#)

Angle tolerance, expressed in the current angle
unit.

[AutoTransitionThickness](#)

Indicates whether the
[EPatternFinder::TransitionThickness](#) property is
automatically computed or not.

[CachedPivot](#)

-

[ContrastMode](#)

Contrast of the instance, as defined in
[EFindContrastMode](#).

FindExtension	Extension value, that is the pattern margin size (in pixels) that is allowed to go out of the search field.
ForcedThreshold	Forced threshold, between [0, 255].
Interpolate	Whether interpolation is performed when searching for a pattern occurrence.
LearningDone	Indicates whether a pattern has already been learnt.
LightBalance	Light balance, between [-1.0, 1.0] .
LocalSearchMode	Sets the local search mode.
MaxFeaturePoints	Maximum number of feature points at the fine stage.
MaxInstances	Maximum number of instances to be found.
MinFeaturePoints	Minimum number of feature points at the coarse stage.
MinScore	Minimum score of found instances, between [-1.0, 1.0] .

PatternType	Pattern type, as defined in EPatternType .
Pivot	Reference point in the model.
ReductionMode	The reduction mode that is to be used when learning the model (automatic or manual), as defined in EReductionMode .
ReductionStrength	The reduction strength that is to be used when learning the model, between 0 and 1 .
Scale	-
ScaleBias	Scale bias, expressed in units (not in percent).
ScaleSearchExtent	The scaling extension of the search neighborhood. See the EPatternFinder::LocalSearchMode property description for further details.
ScaleTolerance	Scale tolerance, expressed in units (not in percent).
Score	-
ThinStructureMode	Mode for EPatternType_ThinStructure , as defined in EThinStructureMode .
TransitionThickness	Transition thickness, expressed in pixels.

Type	Shape type.
XSearchExtent	The X-axis extension of the search neighborhood. See the EPatternFinder::LocalSearchMode property description for further details.
YSearchExtent	M The Y-axis extension of the search neighborhood. See the EPatternFinder::LocalSearchMode property description for further details. E
THODS	
CopyLearntPattern	Copies the learnt pattern in the supplied image. If no pattern has been learned, an exception with code EError_NoPatternLearned will be thrown.
DrawModel	Draws the model features with an overlay in image coordinates.
DrawModelWithCurrentPen	Draws the model features with an overlay in image coordinates.
EPatternFinder	Constructs a EPatternFinder context.
Find	Locates a set of possible occurrences of the learnt pattern in the supplied search field.

Learn

Learns a given pattern and stores it in the [EPatternFinder](#) object.

operator=

Copies all the data from another [EPatternFinder](#) object into the current [EPatternFinder](#) object

P

atternFinder.Angle

-

Angle As Single

read-only

EPatternFinder.AngleBias

Angle bias, expressed in the current angle unit.

[VB6]

AngleBias As Single

read-write

Remarks

The **AngleBias** defines the angle offset between the model and the instances. Finding the pattern is performed in range **AngleBias** +/- [EPatternFinder::AngleTolerance](#). This range should not exceed a full turn. Default: **0.0**.

EPatternFinder.AngleSearchExtent

The angular extension of the search neighborhood. See the [EPatternFinder::LocalSearchMode](#) property description for further details.

[VB6]

AngleSearchExtent As Long

read-write

EPatternFinder.AngleTolerance

Angle tolerance, expressed in the current angle unit.

[VB6]

AngleTolerance As Single

read-write

Remarks

The **AngleTolerance** defines the angle allowance of the instances around the [EPatternFinder::AngleBias](#). Finding the pattern is performed in range **EPatternFinder::AngleBias +/- AngleTolerance**. This range should not exceed a full turn. A **NULL** tolerance can be set, in which case the angle bias value is assumed. Default: **0.0**.

EPatternFinder.AutoTransitionThickness

Indicates whether the [EPatternFinder::TransitionThickness](#) property is automatically computed or not.

[VB6]

AutoTransitionThickness As Boolean

read-write

Remarks

If set to **TRUE**, [EPatternFinder::TransitionThickness](#) is automatically computed during the [EPatternFinder::Learn](#) method call. This computed value will be used (by [EPatternFinder::Find](#)) until a new [EPatternFinder::Learn](#) is done. Even if the user explicitly sets the [EPatternFinder::TransitionThickness](#), it will have no effect, as [EPatternFinder::Find](#) will use the computed value. If set to **FALSE**, [EPatternFinder::TransitionThickness](#) is set by the user. It is never automatically changed by a [EPatternFinder::Learn](#) method call. Default: **TRUE**.

EPatternFinder.CachedPivot

-

[VB6]

CachedPivot As EPoint

read-only

EPatternFinder.ContrastMode

Contrast of the instance, as defined in [EFindContrastMode](#).

[VB6]

ContrastMode As EFindContrastMode

read-write

Remarks

This is a [EPatternType_ConsistentEdges](#) pattern type property. It defines the contrast of regions. Contrast can be normal (as in the model), inverse (inverse contrast of the model), or any (same or inverse contrast of the model). Default: [EFindContrastMode_Normal](#).

EPatternFinder.CopyLearntPattern

Copies the learnt pattern in the supplied image. If no pattern has been learned, an exception with code [EError_NoPatternLearnt](#) will be thrown.

[VB6]

```
void CopyLearntPattern(  
    EImageBW8 image  
)
```

Parameters

image

Pointer to the image in which the learnt pattern will be returned.

EPatternFinder.DrawModel

Draws the model features with an overlay in image coordinates.

[VB6]

```
void DrawModel(  
    Long graphicContext,  
    Single zoomX,  
    Single zoomY,  
    Single panX,  
    Single panY  
)
```

```
void DrawModel(
    Long graphicContext,
    ERGBColor color,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void DrawModel(
    EDrawAdapter graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

graphicContext

Handle to the device context of the destination windows.

zoomX

Horizontal zooming factor.

zoomY

Vertical zooming factor. If set to **0**, the horizontal zooming factor will be used for isotropic zooming.

panX

Horizontal panning offset.

panY

Vertical panning offset.

color

The color in which to draw the overlay.

EPatternFinder.DrawModelWithCurrentPen

Draws the model features with an overlay in image coordinates.

[VB6]

```
void DrawModelWithCurrentPen(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

graphicContext

Handle to the device context of the destination windows.

zoomX

Horizontal zooming factor.

zoomY

Vertical zooming factor. If set to **0**, the horizontal zooming factor will be used for isotropic zooming.

panX

Horizontal panning offset.

panY

Vertical panning offset.

EPatternFinder.EPatternFinder

Constructs a **EPatternFinder** context.

[VB6]

```
void EPatternFinder(
    )

void EPatternFinder(
    EPatternFinder other
)
```

Parameters

other

Another EPatternFinder object to be copied in the new EPatternFinder object.

Remarks

All properties are initialized to their respective default values.

EPatternFinder.Find

Locates a set of possible occurrences of the learnt pattern in the supplied search field.

[VB6]

```
() EFoundPattern Find(  
    EROIBW8 source  
)
```

Parameters

source

Image or part of an image in which the learnt model has to be searched for.

Remarks

This method will fail if no pattern has been learnt previously. The result is a vector of **EFoundPattern** objects.

EPatternFinder.FindExtension

Extension value, that is the pattern margin size (in pixels) that is allowed to go out of the search field.

[VB6]

FindExtension As Long

read-write

Remarks

When a non-**NULL** value is attributed to the extension, the detection of instances partially out of the ROI is allowed. The extension value defines how much the ROI is extended. Default: **0**.

EPatternFinder.ForcedThreshold

Forced threshold, between [0, 255].

[VB6]

ForcedThreshold As Long

read-write

Remarks

This property fixes an absolute gray-level threshold to help the [EPatternFinder](#) in the extraction of regions in the [EPatternType_ContrastingRegions](#). Default: **0**, which means that the thresholding is computed automatically. Once this property has been changed, a new learning process has to be done, to take the new value into account. Note that this property will remain to its value even after a new learning process. An efficient way to see the effect of changing this property is to use the [EPatternFinder::DrawModel](#) method.

EPatternFinder.Interpolate

Whether interpolation is performed when searching for a pattern occurrence.

[VB6]

Interpolate As Boolean

read-write

Remarks

By default, matching is done with a one-pixel precision for all degrees of freedom (translation, rotation and scaling). You can use an additional interpolation process to achieve sub-pixel accuracy. This generally leads to an improvement of the sub-pixel accuracy by a factor larger than 10. This is possible only when the found instances match closely the model. A score higher than 0.99 indicates that the instances are a close match of the model. In other words, the instance is considered to be more accurate when the score is higher. The added computational cost is low. Default: **TRUE**.

EPatternFinder.Learn

Learns a given pattern and stores it in the **EPatternFinder** object.

[VB6]

```
void Learn(
    EROIBW8 pattern,
    EROIBW8 dontCare
)
```

Parameters

pattern

Model to be learnt (ROI).

dontCare

"Don't care" area mask (ROI).

Remarks

Learning another pattern erases the information stored for the first one. A "don't care area" can be set as argument, allowing to mask, while learning, certain parts of the pattern, and do not take them into account. The "don't care area" mask should have the same size than the model.

EPatternFinder.LearningDone

Indicates whether a pattern has already been learnt.

[VB6]

LearningDone As Boolean

read-only

EPatternFinder.LightBalance

Light balance, between **[-1.0, 1.0]**.

[VB6]

LightBalance As Single

read-write

Remarks

Consistent Edges and Thin Structures In the [EPatternType_ConsistentEdges](#) and [EPatternType_ThinStructure](#) modes, the **LightBalance** property governs the selection of the feature points while learning the model. It drives which edge points are eligible as feature points in the model, by defining a criterion for ignoring those edge points that are not sharp enough. As a consequence, this property will influence the spatial distribution of the feature points. In the aforementioned operating modes, the feature points are the places in the image that exhibit a strong variation in the gray level signal. Mathematically, these places are those at which the magnitude of the gradient is significant. The **LightBalance** property defines the way the latter threshold on the magnitude of the gradient is computed, through a careful analysis of the dynamics of gradient. The more the **LightBalance** tends to -1, the more tolerant will be the threshold, and the more edge points will be considered as candidates for becoming feature points. Conversely, as the **LightBalance** property becomes close to 1, only the points with a high gradient magnitude are taken into consideration. In other words, a small **LightBalance** defines a loose criterion for defining what an edge point is, whereas a great value implies a conservative criterion. By default, this property is fixed to **0.0**. This is an appropriate value for most images which are encountered in industrial machine vision. Contrasting Regions When using the [EPatternType_ContrastingRegions](#) pattern type, this property allows the user to compensate for poor lighting conditions of the model. The more the **LightBalance** tends to 1, the lower the threshold will be on the model, and the more dark regions will be considered. Conversely, as the **LightBalance** parameter becomes close to -1, only the bright regions are taken into consideration. The **LightBalance** is automatically set to **0.0** after a learning process. Once the **LightBalance** is changed, a new learning process has to be done to take the new value into account. An efficient way to see the effect of changing this property is to use the [EPatternFinder::DrawModel](#) method.

EPatternFinder.LocalSearchMode

Sets the local search mode.

[VB6]

LocalSearchMode As [ELocalSearchMode](#)

read-write

Remarks

In the multi-stage approach of EasyFind, pattern occurrence candidates are at first found at the coarsest stage. Then, at each of the following stages, their position and score are refined until the last and finest one. This refining is achieved by searching for better candidates in the neighborhood of each of the ones found in the previous stage. The local search mode allows the user to set the extent of this neighborhood. By default, the local search mode is set to [ELocalSearchMode_Basic](#).

EPatternFinder.MaxFeaturePoints

Maximum number of feature points at the fine stage.

[VB6]

MaxFeaturePoints As Long

read-write

Remarks

Default: **1024**. Reserved use.

EPatternFinder.MaxInstances

Maximum number of instances to be found.

[VB6]

MaxInstances As Long

read-write

Remarks

Default: **1**.

EPatternFinder.MinFeaturePoints

Minimum number of feature points at the coarse stage.

[VB6]

MinFeaturePoints As Long

read-write

Remarks

Default: **8**. Reserved use.

EPatternFinder.MinScore

Minimum score of found instances, between **[-1.0, 1.0]**.

[VB6]

MinScore As Single

read-write

Remarks

Instances with a score under the **MinScore** will not be returned.

EPatternFinder.operator=

Copies all the data from another EPatternFinder object into the current EPatternFinder object

[VB6]

```
EPatternFinder operator=(  
    EPatternFinder other  
)
```

Parameters

other

EPatternFinder object to be copied

EPatternFinder.PatternType

Pattern type, as defined in [EPatternType](#).

[VB6]

PatternType As EPatternType

read-write

Remarks

This property informs the [EPatternFinder](#) of the general nature of the model to be learnt.

Default: [EPatternType_ConsistentEdges](#).

EPatternFinder.Pivot

Reference point in the model.

[VB6]

Pivot As EPoint

read-write

Remarks

The coordinates of the reference point are relative to the upper left corner of the model. The location of an instance (Coordinates (X,Y)) is the location of its reference point defined in the model. By default, the pivot is a [EPoint](#) set to the pattern center. [EPoint](#) is a structure that contains two x and y float values.

EPatternFinder.ReductionMode

The reduction mode that is to be used when learning the model (automatic or manual), as defined in [EReductionMode](#).

[VB6]

ReductionMode As EReductionMode

read-write

Remarks

Specifies whether the best-guess method should be used to assert the level of reduction that will be used when learning the model. If this property is set to [EReductionMode_Manual](#), it is up to the user to provide a suitable reduction strength. This value is only used when learning the model. Default: [EReductionMode_Auto](#), which means that the best-guess algorithm is used by default.

EPatternFinder.ReductionStrength

The reduction strength that is to be used when learning the model, between **0** and **1**.

[VB6]

ReductionStrength As Single

read-write

Remarks

Specifies the reduction strength for learning the model (encoded as a percentage). Its precise semantics depends on the reduction mode (see the [EPatternFinder::ReductionMode](#) property):

- * In the automatic reduction mode, its value is undefined until a model is learned. When a model is learned (i.e. after a call to [EPatternFinder::Learn](#)), the value of this property can be read, in which case it reflects the reduction strength that has been automatically chosen by the best-guess algorithm.
- * In the manual reduction mode, this property must be set by the user and is kept constant throughout the entire lifetime of the object. The new value of the property is only used at the following call to [EPatternFinder::Learn](#). This value only has an effect when learning the model. Default: The default value depends on the value of the [EPatternFinder::ReductionMode](#) property. Allowed values: Floating-point number in the interval **[0..1]**.

EPatternFinder.Scale

-

[VB6]

Scale As Single

read-only

EPatternFinder.ScaleBias

Scale bias, expressed in units (not in percent).

[VB6]

ScaleBias As Single

read-write

Remarks

The **ScaleBias** defines the scale factor between the model and the instances. Finding the pattern is performed in range **ScaleBias +/- ScaleTolerance**. This range should not exceed **[0.5..2.5]** (50 % to 250 % scaling). Default: **1.0** (100 %).

EPatternFinder.ScaleSearchExtent

The scaling extension of the search neighborhood. See the [EPatternFinder::LocalSearchMode](#) property description for further details.

[VB6]

ScaleSearchExtent As Long

read-write

EPatternFinder.ScaleTolerance

Scale tolerance, expressed in units (not in percent).

[VB6]

ScaleTolerance As Single

read-write

Remarks

The **ScaleTolerance** defines the scale allowance of the instances around the [EPatternFinder::ScaleBias](#). Finding the pattern is performed in range [EPatternFinder::ScaleBias +/- ScaleTolerance](#). This range should not exceed **[0.5..2]** (50 % to 200 % scaling). A **NULL** tolerance can be set, in which case the scale bias value is assumed. Default: **0.0**.

EPatternFinder.Score

[VB6]

Score As Single

read-only

EPatternFinder.ThinStructureMode

Mode for [EPatternType_ThinStructure](#), as defined in [EThinStructureMode](#).

[VB6]

ThinStructureMode As ETThinStructureMode

read-write

Remarks

[ETThinStructureMode](#) informs EasyFind if thin elements in the model are darker or brighter than regions. Default: [ETThinStructureMode_Auto](#), which detects the best mode between dark or bright.

EPatternFinder.TransitionThickness

Transition thickness, expressed in pixels.

[VB6]

TransitionThickness As Long

read-write

Remarks

This property defines the tolerance on the location of transitions between regions in [EPatternType_ContrastingRegions](#). An efficient way to see the effect of changing this property is to use the [EPatternFinder::DrawModel](#) method. The **TransitionThickness** value, used by [EPatternFinder::Find](#), is either the automatically computed value (by a [EPatternFinder::Learn](#) method call) if [EPatternFinder::AutoTransitionThickness](#) is **TRUE**, or the user-defined value if [EPatternFinder::AutoTransitionThickness](#) is **FALSE**.

Note. If [EPatternFinder::AutoTransitionThickness](#) is **TRUE** and no [EPatternFinder::Learn](#) has been executed, the **TransitionThickness** value that [EPatternFinder::Find](#) will use is undefined, even if it has been manually set in the meantime.

EPatternFinder.Type

Shape type.

[VB6]

Type As EShapeType

read-only

EPatternFinder.XSearchExtent

The X-axis extension of the search neighborhood. See the [EPatternFinder::LocalSearchMode](#) property description for further details.

[VB6]

XSearchExtent As Long

read-write

EPatternFinder.YSearchExtent

The Y-axis extension of the search neighborhood. See the [EPatternFinder::LocalSearchMode](#) property description for further details.

[VB6]

YSearchExtent As Long

read-write

EPeakVector Class

Base Class: [EVector](#)

PROPERTIES

[RawDataPtr](#)

M

Pointer to the vector data.

THODS

[AddElement](#)

Appends (adds at the tail) an element to the vector.

[EPeakVector](#)

Constructs a vector.

[GetElement](#)

Returns the vector element at the given index.

operator[]

Gives access to the vector element at the given index.

operator=

Copies all the data from another EPeakVector object into the current EPeakVector object

SetElement

E Modifies the vector element at the given index by the given value.

P

EPeakVector.AddElement

Appends (adds at the tail) an element to the vector.

[VB6]

```
void AddElement(  
    EPeak element  
)
```

Parameters

element

The element to be added.

EPeakVector.EPeakVector

Constructs a vector.

```
[VB6]
void EPeakVector(
    )
void EPeakVector(
    EPeakVector other
)
void EPeakVector(
    Long maxNumberOfElements
)
```

Parameters

other

EPeakVector object to be copied

maxNumberOfElements

Optionally, memory can be pre-allocated to accommodate a given number of elements.

EPeakVector.GetElement

Returns the vector element at the given index.

```
[VB6]
```

```
EPeak GetElement(
    Long index
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded) of the element to be accessed.

Remarks

If the given index is outside the bounds of the vector, the error code [EError_Parameter1OutOfRange](#) is set.

EPeakVector.operator[]

Gives access to the vector element at the given index.

```
[VB6]  
EPeak operator[](  
    Long index  
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded) of the element to be accessed.

EPeakVector.operator=

Copies all the data from another EPeakVector object into the current EPeakVector object

```
[VB6]  
EPeakVector operator=(  
    EPeakVector other  
)
```

Parameters

other

EPeakVector object to be copied

EPeakVector.R rawDataPtr

Pointer to the vector data.

[VB6]

RawDataPtr As Long

read-only

EPeakVector.SetElement

Modifies the vector element at the given index by the given value.

[VB6]

```
void SetElement(  
    Long index,  
    EPeak value  
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded), of the element to be modified.

value

The new value for the element.

Remarks

If the given index is outside the bounds of the vector, the error code [EError_Parameter1OutOfRange](#) is set.

EPixelAccessor Class

-

Derived Class(es): [EPixelRegion](#) [EPixelContainer](#)

PROPERTIES

Height

-

RowPitch

-

Type

-

Width

M**E****THODS**

GetBufferPtr

-

GetCheckedBufferPtr

-

IsSameType

-

IsVoid

Tests if the EPixelAccessor has a zero size.

P

ixelAccessor.GetBufferPtr

```
[VB6]
Long GetBufferPtr(
)
Long GetBufferPtr(
)
Long GetBufferPtr(
    Long x,
    Long y
)
Long GetBufferPtr(
    Long x,
    Long y
)
```

Parameters

x

-

y

-

EPixelAccessor.GetCheckedBufferPtr

```
-
```

```
[VB6]
Long GetCheckedBufferPtr(
    Long x,
    Long y
)
Long GetCheckedBufferPtr(
    Long x,
    Long y
)
```

Parameters

x

-
Y
-

EPixelAccessor.Height

[VB6]

Height As Long

read-write

EPixelAccessor.IsSameType

[VB6]

```
Boolean IsSameType(  
    EPixelAccessor other  
)
```

Parameters

other

EPixelAccessor.IsVoid

Tests if the EPixelAccessor has a zero size.

[VB6]

```
Boolean IsVoid()  
)
```

EPixelAccessor.RowPitch

-

[VB6]

RowPitch As Long

read-only

EPixelAccessor.Type

-

[VB6]

Type As EImageType

read-only

EPixelAccessor.Width

-

[VB6]

Width As Long

read-write

EPixelContainer Class

Represents a generic pixel container.

Base Class: [EPixelAccessor](#)

Derived Class(es): [EDepthMap](#)

PROPERTIES

[BitsPerPixel](#)

Gets the number of storage bits per pixel.

[Height](#)

Height.

[PlanesPerPixel](#)

Gets the number of color components in each pixel of the ROI/image.

[RowPitch](#)

Buffer row pitch.

[Width](#)

Width.

METHODS

AsEBaseROI	-
CopyTo	-
Draw	Draws a pixel container in a device context.
GetBufferPtr	Retrieves the pointer to the pixel buffer.
GetCheckedBufferPtr	Retrieves the pointer to the pixel buffer.
Load	Restores an image stored in the given file.
operator=	-
Save	Saves the EPixelContainer object to the given file.
SaveJpeg	Saves the EPixelContainer object to the given file, in JPEG format.
SaveJpeg2K	Saves the EPixelContainer object to the given file, in JPEG 2000 format.
SetBufferPtr	Sets the pointer to an externally allocated image buffer.

SetSize

Sets the width and height of a pixel container.

P

ixelContainer.AsEBaseROI

-

[VB6]

```
EBaseROI AsEBaseROI()  
)  
  
EBaseROI AsEBaseROI()  
)
```

EPixelContainer.BitsPerPixel

Gets the number of storage bits per pixel.

[VB6]

```
BitsPerPixel As Long  
read-only
```

EPixelContainer.CopyTo

[VB6]

```
void CopyTo(
    EPixelContainer other
)
```

Parameters

other

-

EPixelContainer.Draw

Draws a a pixel container in a device context.

[VB6]

```
void Draw(
    EDrawAdapter graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void Draw(
    EDrawAdapter graphicContext,
    EC24Vector c24Vector,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void Draw(
    EDrawAdapter graphicContext,
    EBW8Vector bw8Vector,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

```

void Draw(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void Draw(
    Long graphicContext,
    EC24Vector c24Vector,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void Draw(
    Long graphicContext,
    EBW8Vector bw8Vector,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

```

Parameters

graphicContext

Handle to the device context of the destination window.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the image is displayed in **TRUE** scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a **0** value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Pan offset (in pixels) in the horizontal direction. By default, no panning is applied.

panY

Pan offset (in pixels) in the vertical direction. By default, no panning is applied.

c24Vector

When supplied, this parameter allows using a LUT that maps from BW8 to C24 when drawing (false colors).

bw8Vector

When supplied, this parameter allows using a LUT that maps from BW8 to BW8 when drawing.

Remarks

An ROI/image can be drawn (its pixels rendered) using a device context. The horizontal and vertical zooming factors can be different and must be contained in the **1/16..16** range. (MFC users can use the **CDC::GetSafeHdc()** method to obtain a suitable device context handle from a **CDC** instance.)

EPixelContainer.GetBufferPtr

Retrieves the pointer to the pixel buffer.

```
[VB6]
Long GetBufferPtr()
)
Long GetBufferPtr(
    Long x,
    Long y
)
Long GetBufferPtr(
)
Long GetBufferPtr(
    Long x,
    Long y
)
```

Parameters

x

Column of the pixel of which we want the address.

y

Row of the pixel of which we want the address.

Remarks

This function does not check the value of the parameters. Use carefully.

EPixelContainer.GetCheckedBufferPtr

Retrieves the pointer to the pixel buffer.

```
[VB6]  
Long GetCheckedBufferPtr(  
    Long x,  
    Long y  
)  
  
Long GetCheckedBufferPtr(  
    Long x,  
    Long y  
)
```

Parameters

x

Column of the pixel of which we want the address.

y

Row of the pixel of which we want the address.

EPixelContainer.Height

Height.

```
[VB6]
```

Height As Long

read-write

EPixelContainer.Load

Restores an image stored in the given file.

```
[VB6]  
void Load(  
    String path  
)  
  
void Load(  
    ESerializer serializer  
)
```

Parameters

path

Full path of the file.

serializer

The [ESerializer](#) file-like object that is read from.

Remarks

When loading, a depth map is resized if need be. If a serializer is used, then the Euresys proprietary file format is expected. This format preserves attributes and sub-ROIs.

EPixelContainer.operator=

```
[VB6]  
EPixelContainer operator=(  
    EPixelContainer other  
)
```

Parameters

other

EPixelContainer.PlanesPerPixel

Gets the number of color components in each pixel of the ROI/image.

[VB6]

PlanesPerPixel As Long

read-only

EPixelContainer.RowPitch

Buffer row pitch.

[VB6]

RowPitch As Long

read-only

EPixelContainer.Save

Saves the EPixelContainer object to the given file.

[VB6]

```
void Save(  
    String path,  
    EImageFileType type  
)
```

```
void Save(
    ESerializer serializer
)
```

Parameters

path

The full path of the destination file.

type

File format, as defined by [EImageFileType](#). If not specified, the file format is determined from the file extension.

serializer

The [ESerializer](#) file-like object that is written to.

Remarks

By default (if no format is specified), the file format is determined from the file extension. If a serializer is used, then the Euresys proprietary file format is used. This format preserves attributes and sub-ROIs.

EPixelContainer.SaveJpeg

Saves the EPixelContainer object to the given file, in JPEG format.

[VB6]

```
void SaveJpeg(
    String path,
    Long quality
)
```

Parameters

path

The full path of the destination file.

quality

JPEG quality, between 0 and 100 (100 is best quality). The default value is 75.

EPixelContainer.SaveJpeg2K

Saves the EPixelContainer object to the given file, in JPEG 2000 format.

```
[VB6]  
void SaveJpeg2K(  
    String path,  
    Long quality  
)
```

Parameters

path

The full path of the destination file.

quality

JPEG 2000 quality, between 1 and 512. The default value is 16.

EPixelContainer.SetBufferPtr

Sets the pointer to an externally allocated image buffer.

```
[VB6]  
void SetBufferPtr(  
    Long width,  
    Long height,  
    Long imagePointer,  
    Long bitsPerRow  
)
```

Parameters

width

The width of the supplied buffer, in pixels.

height

The height of the supplied buffer, in pixels.

imagePointer

The pointer (aligned on 4 bytes) to the buffer, which must be large enough to hold the image.

bitsPerRow

The total number of bits contained in a row, padding included. Using the value **0** (default) means that this size is computed from the buffer width and the pixel size plus a padding with the smallest possible value that leads to a multiple of 4 bytes (32 bits), which is the minimum padding accepted by [SetBufferPtr](#).

EPixelContainer.SetSize

Sets the width and height of a pixel container.

[VB6]

```
void SetSize(  
    Long width,  
    Long height  
)  
  
void SetSize(  
    EPixelContainer other  
)
```

Parameters

width

The new requested ROI/image width.

height

The new requested ROI/image height.

other

The other ROI/image whose dimensions have to be used for the current object.

Remarks

Open eVision will allocate a new image buffer (deallocate the old image buffer) if the supplied width and height are different from the existing ones. If an external buffer has been specified by means of **SetImagePtr**, it will be kept only if the size does not change. Creating a new Open eVision image buffer and setting its size creates a 4-byte aligned buffer, by default. The *size of an image* is specified as a number of columns (width) and rows (height). The maximum image dimensions are 32767 by 32767. Furthermore, it must fit into the available memory, that depends upon the physical memory, the operating system and the memory already allocated by the process in other modules or libraries. The *placement of an ROI* is given by the x and y coordinates of its upper left pixel relative to its parent image, and also by its width and its height.

EPixelContainer.Width

Width.

[VB6]

Width As Long

read-write

EPixelRegion Class

Base Class: [EPixelAccessor](#)

Derived Class(es): [EROI](#)

EPoint Class

An exact (floating-point) location in the 2D space.

Derived Class(es): [EFrame](#)

PROPERTIES

Center Center coordinates of a EPoint object.

X Abscissa (X coordinate) of the EPoint object

Y **M** Ordinate (Y coordinate) of the EPoint object

THODS

Area Compute the oriented area of the parallelogram built on two [EPoint](#).

Argument Compute the polar argument of a EPoint object.

CopyTo Copies all the data of the current EPoint object into another EPoint object and returns it.

Distance Returns the distance between the addressed point and an EPoint object.

Dot Compute the dot product of two EPoint object.

EPoint Constructs a EPoint object.

MidPoint**Modulus**

Returns the middle coordinate between this EPoint object and another EPoint object.

operator-

Subtracts from the current EPoint center coordinates the center coordinates of another EPoint object.

operator!=

Compares the current EPoint center coordinates with the center coordinates of another EPoint object.

operator*

Multiplies the current EPoint center coordinates by a given multiplier.

operator/

Divides the current EPoint center coordinates by a given divisor.

operator+

Adds to the current EPoint center coordinates the center coordinates of another EPoint object.

operator=

Copies all the data from another EPoint object into the current EPoint object.

operator==

Compares the current EPoint center coordinates with the center coordinates of another EPoint object.

Project

Compute the orthogonal projection of a EPoint on another shape.

Rotate

Returns another EPoint object containing the coordinated of the rotated point.

SetCenterXY

Sets the center coordinates of a EPoint object.

Square

Compute the sum of the squared coordinates of a EPoint.

SquaredDistance

Compute the squared distance between two EPoint.

P

oint.Area

Compute the oriented area of the parallelogram built on two EPoint.

[VB6]

```
Single Area(
    EPoint Point
)
```

Parameters

Point

Second edge of the parallelogram.

Remarks

Compute the oriented area of the parallelogram built on two [EPoint](#). The area is counted as positive if the oriented vector pair ('first edge', 'second edge') is in the same sense that the axis frame. This oriented area can also be viewed as the z-coordinate of a vector product of two 3D vectors obtained in supplementing each edges with a third z-coordinate (setted to zero).

EPoint.Argument

Compute the polar argument of a EPoint object.

[VB6]

```
Single Argument()  
)
```

Remarks

Compute the angle (in radians) between the oriented X-axis and the vector going from the axis origin and the [EPoint](#). If the axis frame is orthogonal, this number is also the polar argument of the [EPoint](#).

EPoint.Center

Center coordinates of a EPoint object.

[VB6]

```
Center As EPoint  
read-write
```

EPoint.CopyTo

Copies all the data of the current EPoint object into another EPoint object and returns it.

[VB6]

```
EPoint CopyTo(  
    EPoint other  
)
```

Parameters

other

Pointer to the EPoint object in which the current EPoint object data have to be copied.

Remarks

In case of a **NULL** pointer, a new EPoint object will be created and returned.

EPoint.Distance

Returns the distance between the addressed point and an EPoint object.

[VB6]

```
Single Distance(  
    EPoint point  
)  
  
Single Distance(  
    ELine line,  
    Boolean segmentOnly  
)  
  
Single Distance(  
    ECircle circle,  
    Boolean arcOnly  
)
```

Parameters

point

[EPoint](#) object with which to calculate the distance.

line

[ELine](#) object with which to calculate the distance.

segmentOnly

By default (**FALSE**), the line is not restricted to a segment.

circle

[ECircle](#) object with which to calculate the distance.

arcOnly

By default (**FALSE**), the circle is not restricted to an arc.

Remarks

Many EasyGauge members provide measurement result as a [EPoint](#) object (see [EPointGauge::Center](#), [EPointGauge::GetMeasuredPoint](#),...). The EPoint class has its own members to retrieve all the information pertaining to a point. Among them, the **Distance** method returns the distance between a point pair or between a point and a line segment, a circle arc or a rectangle.

EPoint.Dot

Compute the dot product of two EPoint object.

[VB6]

```
Single Dot(  
    EPoint Point  
)
```

Parameters

Point

Second factor of the dot product.

EPoint.EPoint

Constructs a EPoint object.

```
[VB6]
void EPoint(
    )
void EPoint(
    Single centerX,
    Single centerY
    )
void EPoint(
    EPoint other
    )
```

Parameters

centerX

Center coordinates of the EPoint object.

centerY

Center coordinates of the EPoint object.

other

Another EPoint object to be copied in the new EPoint object.

EPoint.MidPoint

Returns the middle coordinate between this EPoint object and another EPoint object.

```
[VB6]
EPoint MidPoint(
    EPoint Point
    )
```

Parameters

Point

The other EPoint object.

EPoint.Modulus

Compute the euclidian modulus of a [EPoint](#).

[VB6]

```
Single Modulus(  
    )
```

Remarks

Compute the squared root of the sum of the squared coordinates of an [EPoint](#). If the axis frame is orthogonal, this number is also the euclidian norm of the [EPoint](#).

EPoint.operator-

Subtracts from the current EPoint center coordinates the center coordinates of another EPoint object.

[VB6]

```
EPoint operator-(  
    EPoint point  
    )
```

Parameters

point

The other EPoint object.

EPoint.operator!=

Compares the current EPoint center coordinates with the center coordinates of another EPoint object.

[VB6]

```
Boolean operator!=(
    EPoint point
)
```

Parameters

point

The other EPoint object.

Remarks

Returns **TRUE** if `EPoint::X` or `EPoint::Y` are respectively different.

EPoint.operator*

Multiplies the current EPoint center coordinates by a given multiplier.

[VB6]

```
EPoint operator*(
    Single scalar
)
```

Parameters

scalar

The multiplier.

EPoint.operator/

Divides the current EPoint center coordinates by a given divisor.

[VB6]

```
EPoint operator/(
    Single scalar
)
```

Parameters

scalar

The divisor.

EPoint.operator+

Adds to the current EPoint center coordinates the center coordinates of another EPoint object.

[VB6]

```
EPoint operator+(
    EPoint point
)
```

Parameters

point

The other EPoint object.

EPoint.operator=

Copies all the data from another EPoint object into the current EPoint object.

[VB6]

```
EPoint operator=((
    EPoint other
))
```

Parameters

other

EPoint object to be copied.

EPoint.operator==

Compares the current EPoint center coordinates with the center coordinates of another EPoint object.

[VB6]

```
Boolean operator==(  
    EPoint point  
)
```

Parameters

point

The other EPoint object.

Remarks

Returns **TRUE** if both `EPoint::X` and `EPoint::Y` are respectively the same.

EPoint.Project

Compute the orthogonal projection of a EPoint on another shape.

[VB6]

```
EPoint Project(  
    ELine shape  
)  
  
EPoint Project(  
    ECircle shape  
)
```

Parameters*shape*

Shape object to which point is projected

Remarks

Compute the orthogonal projection of a EPoint on another shape. This computation is only valid when the axis frame is orthogonal.

EPoint.Rotate

Returns another EPoint object containing the coordinates of the rotated point.

[VB6]

```
EPoint Rotate(  
    Single angle  
)
```

Parameters*angle*

Rotation angle (in radians)

Remarks

Rotates a EPoint around the origin **(0, 0)** by an angle of **angle** radians. By definition, the smallest (in absolute value) rotation of the oriented X-Axis toward the oriented Y-Axis is chosen as the positive sense of rotation. In a direct frame, this is also the trigonometric sense (counter clockwise).

EPoint.SetCenterXY

Sets the center coordinates of a EPoint object.

[VB6]

```
void SetCenterXY(  
    Single centerX,  
    Single centerY  
)
```

Parameters

centerX

Center coordinates of the EPoint object.

centerY

Center coordinates of the EPoint object.

EPoint.Square

Compute the sum of the squared coordinates of a [EPoint](#).

[VB6]

```
Single Square(  
)
```

Remarks

Compute the sum of the squared coordinates of a [EPoint](#). If the axis frame is orthogonal, this sum of squares is also the squared euclidian norm of the EPoint object.

EPoint.SquaredDistance

Compute the squared distance between two [EPoint](#).

[VB6]

```
Single SquaredDistance(  
    EPoint Point  
)
```

Parameters

Point
Second [EPoint](#).

Remarks

Compute the sum of squared coordinates differences of two [EPoint](#). If the axis frame is orthogonal, this number is also the squared euclidian distance between two [EPoint](#).

EPoint.X

Abscissa (X coordinate) of the EPoint object

[VB6]
X As Single
read-only

EPoint.Y

Ordinate (Y coordinate) of the EPoint object

[VB6]
Y As Single
read-only

EPointGauge Class

Manages a point location gauge.

Base Class: [EPointShape](#)

PROPERTIES

Active

Sets the flag indicating whether the gauge is active or not.

Center

-

HVConstraint

Status of the restriction on the orientation of the point location gauge or model fitting sample paths.

MinAmplitude

Offset added to the **Threshold** when a peak is to be detected.

MinArea

Minimum area value.

NumMeasuredPoints

Number of edge-crossing points along the point location gauge.

RectangularSamplingArea

Shape	Flag indicating whether the sampling area remains a rectangle when rotated, instead of becoming a parallelogram.
Smoothing	-
Thickness	Number of pixels used for the low-pass filtering operation.
Threshold	Number of parallel segments used to extract the data profile.
Tolerance	Threshold level used to delimit significant peaks in the data profile.
ToleranceAngle	Half length of the point location gauge.
TransitionChoice	Rotation angle of the point location gauge.
TransitionIndex	Transition choice.
	Index (from 0 on) of the transition to be retained when the transition choice parameter is set to ETransitionChoice_NthFromBegin or ETransitionChoice_NthFromEnd .

TransitionType	Transition type.
Type	Shape type.
Valid	M Flag indicating if at least one valid transition has been found. E
THODS	
CopyTo	Copies all the data of the current EPointGauge object into another EPointGauge object, and returns it.
Drag	Moves a handle to a new position and updates the position parameters of the gauge.
Draw	Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode .
DrawWithCurrentPen	Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode .
EPointGauge	Constructs a point measurement context.

GetMeasuredPeak	Returns information pertaining to the derivative peak associated with the specified edge-crossing point, such as its area, amplitude, start, length and center.
GetMeasuredPoint	Returns the coordinates of an edge-crossing point, measured along the unique sample path of the point location gauge.
HitTest	Checks whether the cursor is positioned over a handle (TRUE) or not (FALSE).
Measure	Triggers the point location or the model fitting operation.
operator=	Copies all the data from another EPointGauge object into the current EPointGauge object
Plot	Draws the profile that is crossed by a point location gauge, as defined by EPlotItem .
PlotWithCurrentPen	Draws the profile that is crossed by a point location gauge, as defined by EPlotItem .
Process	Triggers the process pertaining to a shape or gauge and all the daughter gauges attach to it.

SetCenterXY	Sets the center coordinates of a EPointGauge object.
SetTolerances	<p>Sets the half length and the rotation angle of the point location gauge.</p>

ointGauge.Active

Sets the flag indicating whether the gauge is active or not.

Active As Boolean

read-write

Remarks

When complex gauging is required, several gauges can be grouped together. Applying [EPointGauge::Process](#) to the mother gauge or shape triggers the measurement of the whole. Only the active gauges will participate in the process. By default, the gauge is active (**TRUE**).

EPointGauge.Center

Center As EPoint

read-write

EPointGauge.CopyTo

Copies all the data of the current EPointGauge object into another EPointGauge object, and returns it.

[VB6]

```
EPointGauge CopyTo(  
    EPointGauge other,  
    Boolean recursive  
)
```

Parameters

other

Pointer to the EPointGauge object in which the current EPointGauge object data have to be copied.

recursive

TRUE if the children gauges have to be copied as well, **FALSE** otherwise.

Remarks

In case of a **NULL** pointer, a new EPointGauge object will be created and returned.

EPointGauge.Drag

Moves a handle to a new position and updates the position parameters of the gauge.

[VB6]

```
void Drag(  
    Long x,  
    Long y  
)
```

Parameters

x

Cursor current coordinates.

y

Cursor current coordinates.

EPointGauge.Draw

Draws a graphical representation of a point location or model fitting gauge, as defined by [EDrawingMode](#).

[VB6]

```
void Draw(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)

void Draw(
    Long graphicContext,
    ERGBColor color,
    EDrawingMode drawingMode,
    Boolean daughters
)

void Draw(
    EDrawAdapter graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by [EDrawingMode](#).

daughters

TRUE if the daughters gauges are to be displayed also.

color

The color in which to draw the overlay.

EPointGauge.DrawWithCurrentPen

Draws a graphical representation of a point location or model fitting gauge, as defined by [EDrawingMode](#).

[VB6]

```
void DrawWithCurrentPen(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by [EDrawingMode](#).

daughters

TRUE if the daughters gauges are to be displayed also.

EPointGauge.EPointGauge

Constructs a point measurement context.

[VB6]

```
void EPointGauge(
)
void EPointGauge(
    Single centerX,
    Single centerY
)
```

```
void EPointGauge(
    EPointGauge other
)
```

Parameters*centerX*

Point coordinates.

centerY

-

other

Another EPointGauge object to be copied in the new EPointGauge object.

Remarks

With the default constructor, all the parameters are initialized to their respective default values. With the copy constructor, the constructed point measurement context is based on a pre-existing [EPointGauge](#) object. The gauge children are also copied. Hierarchy copying through a copy constructor is always recursive. To disable this recursion, use instead the [EPointGauge::CopyTo](#) method.

EPointGauge.GetMeasuredPeak

Returns information pertaining to the derivative peak associated with the specified edge-crossing point, such as its area, amplitude, start, length and center.

[VB6]

```
EPeak GetMeasuredPeak (
    Long index
)
```

Parameters*index*

Index of the edge-crossing point along the probed line segment, between **0** and [EPointGauge::NumMeasuredPoints](#) (excluded).

Remarks

If **index** is left unchanged from its default value (i.e. **-0 = 0xFFFFFFFF**), the peak associated to the default edge-crossing point is inspected. This default point is chosen according to the transition choice parameter that is managed by the [EPointGauge::TransitionChoice](#) property.

Note. For this method to succeed, it is necessary to previously call [EPointGauge::Measure](#).

EPointGauge.GetMeasuredPoint

Returns the coordinates of an edge-crossing point, measured along the unique sample path of the point location gauge.

[VB6]

```
EPoint GetMeasuredPoint(
    Long index
)
```

Parameters

index

Index of the edge-crossing point along the probed line segment, between **0** and [EPointGauge::NumMeasuredPoints](#) (excluded).

Remarks

These coordinates pertain to the World space; they are expressed in the reference frame to which the current EPointGauge object belongs. An EPointGauge object features only one sample path, which contrasts with the other kinds of gauges. The argument **index** specifies the index of the edge-crossing point that is considered along this unique sample path. If **index** is left unchanged from its default value (i.e. **-0= 0xFFFFFFFF**), the default edge-crossing point is inspected. This default point is chosen according to the transition choice parameter that is managed by the [EPointGauge::TransitionChoice](#) property.

Note. For this method to succeed, it is necessary to previously call [EPointGauge::Measure](#).

EPointGauge.HitTest

Checks whether the cursor is positioned over a handle (**TRUE**) or not (**FALSE**).

[VB6]

```
Boolean HitTest(
    Boolean daughters
)
```

Parameters

daughters

TRUE if the daughters gauges handles have to be considered as well.

EPointGauge.HVConstraint

Status of the restriction on the orientation of the point location gauge or model fitting sample paths.

[VB6]

```
HVConstraint As Boolean
read-write
```

Remarks

Sample paths are the point location gauges placed along the model to be fitted.

EPointGauge.Measure

Triggers the point location or the model fitting operation.

[VB6]

```
void Measure(
    EROIBW8 sourceImage
)
```

Parameters*sourceImage*

Pointer to the source image.

Remarks

When this method is called, and if not enough valid points are detected, then the method returns directly, and the measured gauge is set to the nominal parameters.

EPointGauge.MinAmplitude

Offset added to the **Threshold** when a peak is to be detected.

[VB6]

MinAmplitude As Long

read-write

Remarks

When analyzing a derivative profile, a peak is made up of consecutive pixel values above **Threshold**. To detect weak [strong] transitions, lower [raise] the **Threshold** value. To avoid interference of noise, an additional parameter is provided. The **MinAmplitude** parameter is an offset added to **Threshold** when a peak is to be detected. When the pixel values of the derivative profile do not reach **Threshold + MinAmplitude**, the peak is not taken into account. Anyway, when a peak is taken into account, all the pixels with values above **Threshold** are considered (for more accuracy). Setting the **MinAmplitude** value to **0** merely cancels its effect.

EPointGauge.MinArea

Minimum area value.

[VB6]

MinArea As Long

read-write

Remarks

A transition is detected if its derivative peak reaches **Threshold + MinAmplitude** value, and then declared valid if the area between the peak curve and the horizontal at level **Threshold** reaches the **MinArea** value.

EPointGauge.NumMeasuredPoints

Number of edge-crossing points along the point location gauge.

[VB6]

NumMeasuredPoints As Long

read-only

EPointGauge.operator=

Copies all the data from another EPointGauge object into the current EPointGauge object

[VB6]

```
EPointGauge operator=(  
    EPointGauge other  
)
```

Parameters

other

EPointGauge object to be copied

EPointGauge.Plot

Draws the profile that is crossed by a point location gauge, as defined by [EPlotItem](#).

[VB6]

```
void Plot(
    Long graphicContext,
    EPlotItem drawItems,
    Single width,
    Single height,
    Single originX,
    Single originY
)

void Plot(
    Long graphicContext,
    ERGBColor color,
    EPlotItem drawItems,
    Single width,
    Single height,
    Single originX,
    Single originY
)

void Plot(
    EDrawAdapter graphicContext,
    EPlotItem drawItems,
    Single width,
    Single height,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawItems

Boolean combination of [EPlotItem](#) members, that indicates which items must be displayed.

width

Width of the plot.

height

Height of the plot.

originX

Origin point coordinates of the plot along the X axis.

originY

Origin point coordinates of the plot along the Y axis.

color

The color in which to draw the overlay.

EPointGauge.PlotWithCurrentPen

Draws the profile that is crossed by a point location gauge, as defined by [EPlotItem](#).

[VB6]

```
void PlotWithCurrentPen(
    Long graphicContext,
    EPlotItem drawItems,
    Single width,
    Single height,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawItems

Boolean combination of [EPlotItem](#) members, that indicates which items must be displayed.

width

Width of the plot.

height

Height of the plot.

originX

Origin point coordinates of the plot along the X axis.

originY

Origin point coordinates of the plot along the Y axis.

EPointGauge.Process

Triggers the process pertaining to a shape or gauge and all the daughter gauges attach to it.

[VB6]

```
void Process(
    EROIBW8 sourceImage,
    Boolean daughters
)
```

Parameters

sourceImage

Pointer to the source image.

daughters

Flag indicating whether the daughters shapes inherit of the same behavior.

Remarks

When complex gauging is required, several gauges can be grouped together. Applying **Process** to the mother gauge or shape triggers the measurement of the whole. Only the active gauges will participate in the process.

EPointGauge.RectangularSamplingArea

Flag indicating whether the sampling area remains a rectangle when rotated, instead of becoming a parallelogram.

[VB6]

```
RectangularSamplingArea As Boolean
```

read-write

Remarks

By default, this flag is set to **TRUE**: the sampling area always remains a rectangle. Setting this property is only useful when the thickness transition parameter is greater than 1. In fact, when thickness transition parameter is equal to 1, rectangle and parallelogram reduce to the same segment.

EPointGauge.SetCenterXY

Sets the center coordinates of a EPointGauge object.

[VB6]

```
void SetCenterXY(  
    Single centerX,  
    Single centerY  
)
```

Parameters

centerX

Center coordinates of the EPointGauge object.

centerY

Center coordinates of the EPointGauge object.

EPointGauge.SetTolerances

Sets the half length and the rotation angle of the point location gauge.

[VB6]

```
void SetTolerances(  
    Single tolerance,  
    Single angle  
)
```

Parameters

tolerance

Half length of the point location gauge. The default value is **10**.

angle

Rotation angle of the point location gauge. The default value is **0**.

Remarks

By default, the point location gauge length value is 20 (2x10), which means 20 pixels when the field of view is not calibrated and 20 "units" in case of a calibrated field of view. The sign of the rotation angle depends whether the field of view is calibrated or not. * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value. * When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

EPointGauge.Shape

-

[VB6]

Shape As EPoint

read-only

EPointGauge.Smoothing

Number of pixels used for the low-pass filtering operation.

[VB6]

Smoothing As Long

read-write

Remarks

To reduce the effect of noise, the profile data can be low-pass filtered along the point location gauge direction.

EPointGauge.Thickness

Number of parallel segments used to extract the data profile.

[VB6]

Thickness As Long

read-write

Remarks

To reduce the effect of noise and/or strengthen a transition, several parallel profiles can be accumulated.

EPointGauge.Threshold

Threshold level used to delimit significant peaks in the data profile.

[VB6]

Threshold As Long

read-write

Remarks

When analyzing a derivative profile, a peak is made up of consecutive pixel values above **Threshold**. To detect weak [strong] transitions, lower [raise] the **Threshold** value. To avoid interference of noise, an additional parameter is provided. The **MinAmplitude** parameter is an offset added to **Threshold** when a peak is to be detected. When the pixel values of the derivative profile do not reach **Threshold + MinAmplitude**, the peak is not taken into account. Anyway, when a peak is taken into account, all the pixels with values above **Threshold** are considered (for more accuracy). Setting the **MinAmplitude** value to **0** merely cancels its effect.

EPointGauge.Tolerance

Half length of the point location gauge.

[VB6]

Tolerance As Single

read-write

Remarks

By default, the length of the point location gauge is 20 (2x10), which means 20 pixels when the field of view is not calibrated, and 20 physical units in case of a calibrated field of view.

EPointGauge.ToleranceAngle

Rotation angle of the point location gauge.

[VB6]

ToleranceAngle As Single

read-write

Remarks

By default, the rotation angle of the point location gauge is **0**. The sign of the rotation angle depends whether the field of view is calibrated or not. * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value. * When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

EPointGauge.TransitionChoice

Transition choice.

[VB6]

TransitionChoice As ETransitionChoice

read-write

Remarks

Several peaks may be detected along a point location gauge. This parameter helps to select the desired transition. In case of [ETransitionChoice_NthFromBegin](#) or [ETransitionChoice_NthFromEnd](#) transition choice, set [EPointGauge::TransitionIndex](#) to specify the desired transition. By default, the selected transition corresponds to the one with the largest amplitude ([ETransitionChoice_LargestAmplitude](#)).

EPointGauge.TransitionIndex

Index (from **0** on) of the transition to be retained when the transition choice parameter is set to [ETransitionChoice_NthFromBegin](#) or [ETransitionChoice_NthFromEnd](#).

[VB6]

TransitionIndex As Long

read-write

Remarks

Several peaks may be detected along a point location gauge. This parameter helps to select the desired transition. By default, the first transition is retained (the index value is **0**).

EPointGauge.TransitionType

Transition type.

[VB6]

TransitionType As ETransitionType

read-write

Remarks

The type of a transition tells whether it crosses increasing or decreasing gray-level values. This helps discriminate between nearby edges of an object. By default, the searched transition type is indifferently a black to white or a white to black transition ([ETransitionType_BwOrWb](#)).

EPointGauge.Type

Shape type.

[VB6]

Type As EShapeType

read-only

EPointGauge.Valid

Flag indicating if at least one valid transition has been found.

[VB6]

Valid As Boolean

read-only

Remarks

A **FALSE** value means that no measurement has been performed. A **TRUE** value means that a transition was found along the sample path defined by the [EPointGauge](#), and thus a point was measured.

EPointShape Class

Base Class: [EShape](#)

Derived Class(es): [EPatternFinder](#) [EPointGauge](#)

PROPERTIES

Angle



Center



CenterX



CenterY



Scale



Type	M Shape type.
THODS	
Closest	Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use EShape::ClosestShape .
CopyTo	-
Drag	-
Draw	Draws a graphical representation of a shape, as defined by EDrawingMode .
DrawWithCurrentPen	Draws a graphical representation of a shape, as defined by EDrawingMode .
HitTest	-
operator!=	-
operator=	Copies all the data from another EPointShape object into the current EPointShape object
operator==	Copies all the data from another EPointShape object into the current EPointShape object

SetCenterXY

Sets the center coordinates of a EPointShape object.

P

EPointShape.Angle

-

[VB6]

Angle As Single

read-write

EPointShape.Center

-

[VB6]

Center As EPoint

read-write

EPointShape.CenterX

-

[VB6]

CenterX As Single

read-only

EPointShape.CenterY

-

[VB6]

CenterY As Single

read-only

EPointShape.Closest

Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use [EShape::ClosestShape](#).

[VB6]

```
void Closest()  
)
```

EPointShape.CopyTo

-

[VB6]

```
EPointShape CopyTo(  
    EPointShape dest,  
    Boolean bRecursive  
)
```

Parameters

dest
-
bRecursive
-

EPointShape.Drag

```
[VB6]  
void Drag(  
    Long n32CursorX,  
    Long n32CursorY  
)
```

Parameters

n32CursorX
-
n32CursorY
-

EPointShape.Draw

Draws a graphical representation of a shape, as defined by [EDrawingMode](#).

```
[VB6]

void Draw(
    EDrawAdapter graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)

void Draw(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)

void Draw(
    Long graphicContext,
    ERGBColor color,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by [EDrawingMode](#).

daughters

TRUE if the daughters gauges are to be displayed also.

color

EPointShape.DrawWithCurrentPen

Draws a graphical representation of a shape, as defined by [EDrawingMode](#).

```
[VB6]
```

```
void DrawWithCurrentPen(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by [EDrawingMode](#).

daughters

TRUE if the daughters gauges are to be displayed also.

EPointShape.HitTest

-

[VB6]

```
Boolean HitTest(
    Boolean bDaughters
)
```

Parameters

bDaughters

-

EPointShape.operator!=

-

[VB6]

```
Boolean operator!=(
    EPointShape other
)
```

Parameters

other

-

EPointShape.operator=

Copies all the data from another EPointShape object into the current EPointShape object

[VB6]

```
EPointShape operator=( 
    EPointShape other
)
```

Parameters

other

EPointShape object to be copied

EPointShape.operator==

Copies all the data from another EPointShape object into the current EPointShape object

[VB6]

```
Boolean operator==( 
    EPointShape other
)
```

Parameters

other

EPointShape object to be copied

EPointShape.Scale

-

[VB6]

Scale As Single

read-write

EPointShape.SetCenterXY

Sets the center coordinates of a EPointShape object.

[VB6]

```
void SetCenterXY(  
    Single centerX,  
    Single centerY  
)
```

Parameters

centerX

Center coordinates of the EPointShape object.

centerY

Center coordinates of the EPointShape object.

EPointShape.Type

Shape type.

[VB6]

Type As EShapeType

read-only

EPseudoColorLookup Class

Describes a lookup table, that is used to for pseudo-coloring (i.e. for assigning colors to gray-level images).

METHODS

[EPseudoColorLookup](#)

Default constructor of EPseudoColorLookup objects.

[SetShading](#)

Sets up a pseudo-color mapping such that gray level **0** corresponds to color **c24Black**, gray level **255** corresponds to color **c24White**, and intermediate values are interpolated linearly between these two extremes.

seudoColorLookup.E PseudoColorLookup

Default constructor of EPseudoColorLookup objects.

```
[VB6]
void EPseudoColorLookup(
    EPseudoColorLookup other
)
void EPseudoColorLookup(
)
```

Parameters*other*

EPseudoColorLookup.SetShading

Sets up a pseudo-color mapping such that gray level **0** corresponds to color **c24Black**, gray level **255** corresponds to color **c24White**, and intermediate values are interpolated linearly between these two extremes.

```
[VB6]
void SetShading(
    EC24 black,
    EC24 white,
    EColorSystem colorSystem,
    Boolean wrap
)
```

Parameters*black*

Color to be mapped on a black (value **0**) pixel.

white

Color to be mapped on a white (value **255**) pixel.

colorSystem

Color system in which interpolation takes place.

wrap

If the color system supports a hue component, indicates whether hue wrap around must be applied.

Remarks

Furthermore, interpolation is performed in the designated color system. Even though interpolation is performed in an arbitrary color system, the extreme colors are specified in the RGB space. To obtain interesting shades of colors, it is recommended to interpolate on the hue component alone.

EQRCode Class

Represents a QR code found in the search field.

PROPERTIES

DecodedStream	Decoded stream.
Geometry	Geometry of the QR code.
IsDecodingReliable	Decoding reliability.
Level	Level of the QR code.
Model	Model of the QR code.
UnusedErrorCorrection	Unused error correction.
Version	Version of the QR code.

METHODS

Draw

Draws the QR code using a pre-defined pen.

DrawWithCurrentPen

Draws the QR code using the pen currently set in the graphical context.

EQRCode

Creates an EQRCode object.

operator=

E-

Q

RCode.DecodedStream

Decoded stream.

[VB6]

DecodedStream As EQRCodeDecodedStream

read-only

EQRCode.Draw

Draws the QR code using a pre-defined pen.

[VB6]

```
void Draw(
    Long hDC,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

hDC

-

zoomX

-

zoomY

-

panX

-

panY

-

EQRCode.DrawWithCurrentPen

Draws the QR code using the pen currently set in the graphical context.

[VB6]

```
void DrawWithCurrentPen(
    Long hDC,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

hDC

-

zoomX

-

zoomY

-

panX

-

panY

-

EQRCode.EQRCode

Creates an EQRCode object.

[VB6]

```
void EQRCode()
)

void EQRCode(
    EQRCode other
)
```

Parameters

other

-

EQRCode.Geometry

Geometry of the QR code.

[VB6]

Geometry As EQRCodeGeometry

read-only

EQRCode.IsDecodingReliable

Decoding reliability.

[VB6]

IsDecodingReliable As Boolean

read-only

EQRCode.Level

Level of the QR code.

[VB6]

Level As EQRCodeLevel

read-only

EQRCode.Model

Model of the QR code.

[VB6]

Model As EQRCodeModel

read-only

EQRCode.operator=

-

[VB6]

```
EQRCode operator=(  
    EQRCode other  
)
```

Parameters

other

-

EQRCode.UnusedErrorCorrection

Unused error correction.

[VB6]

```
UnusedErrorCorrection As Single  
read-only
```

Remarks

Returns the amount of unused error correction as a percentage. This parameter ranges from 0 to 1. Returns -1 if error correction failed (too many errors).

EQRCode.Version

Version of the QR code.

[VB6]

Version As Long

read-only

EQRCodeDecodedStream Class

Represents the complete decoded stream extracted from a QR code.

PROPERTIES

ApplicationIndicator Application indicator.

CodingMode Coding mode.

DecodedStreamParts Decoded stream parts.

RawBitstream **M** Raw bit stream.

E

THODS

EQRCodeDecodedStream Creates an EQRCodeDecodedStream object.

operator= -

EQRCodeDecodedStream.ApplicationIndicator

Application indicator.

[VB6]

ApplicationIndicator As Long

read-only

Remarks

The application indicator is relevant if the coding mode of the QR code is FNC1/AIM only.

EQRCodeDecodedStream.CodingMode

Coding mode.

[VB6]

CodingMode As EQRCodeCodingMode

read-only

EQRCodeDecodedStream.DecodedStreamParts

Decoded stream parts.

[VB6]

DecodedStreamParts As ()EQRCodeDecodedStreamPart

read-only

EQRCodeDecodedStream.EQRCodeDecodedStream

Creates an EQRCodeDecodedStream object.

```
[VB6]  
void EQRCodeDecodedStream()  
  
void EQRCodeDecodedStream(  
    EQRCodeDecodedStream other  
)
```

Parameters

other

EQRCodeDecodedStream.operator=

```
[VB6]  
EQRCodeDecodedStream operator=(  
    EQRCodeDecodedStream other  
)
```

Parameters

other

EQRCodeDecodedStream.RawBitstream

Raw bit stream.

[VB6]

RawBitstream As ()Byte

read-only

Remarks

The raw bit stream is the bit stream of the QR code after unmasking and error correction, but before decoding.

EQRCodeDecodedStreamPart Class

Represents part of a decoded stream extracted from a QR code.

PROPERTIES

DecodedData

Decoded data.

Encoding

M Encoding.

E

THODS

EQRCodeDecodedStreamPart

Creates an **EQRCodeDecodedStreamPart** object.

operator=

-

EQRCodeDecodedStreamPart.DecodedData

Decoded data.

[VB6]

DecodedData As ()Byte

read-only

EQRCodeDecodedStreamPart.Encoding

Encoding.

[VB6]

Encoding As EQRCodeEncoding

read-only

EQRCodeDecodedStreamPart.EQRCodeDecodedSt reamPart

Creates an [EQRCodeDecodedStreamPart](#) object.

[VB6]

```
void EQRCodeDecodedStreamPart()  
)
```

```
void EQRCodeDecodedStreamPart(
    EQRCodeDecodedStreamPart other
)
```

Parameters

other

-

`EQRCodeDecodedStreamPart.operator=`

[VB6]

```
EQRCodeDecodedStreamPart operator=(
    EQRCodeDecodedStreamPart other
)
```

Parameters

other

-

EQRCodeGeometry Class

Represents the geometry of a QR code.

PROPERTIES

`FinderPatternCenters`

Finder patterns centers.

Position

M Position of the QR code.

THODS**Draw**

Draws the QR code geometry using a pre-defined pen.

DrawWithCurrentPen

Draws the QR code geometry using the pen currently set in the graphical context.

QRCodeGeometry

Creates an [QRCodeGeometry](#) object.

operator=

E-

Q

RCodeGeometry.Draw

Draws the QR code geometry using a pre-defined pen.

[VB6]

```
void Draw(
    Long hDC,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)
```

Parameters

hDC

-
zoomX

-
zoomY

-
panX

-
panY

EQRCodeGeometry.DrawWithCurrentPen

Draws the QR code geometry using the pen currently set in the graphical context.

[VB6]

```
void DrawWithCurrentPen(  
    Long hDC,  
    Single zoomX,  
    Single zoomY,  
    Single panX,  
    Single panY  
)
```

Parameters

hDC

-
zoomX

-
zoomY

-
panX

-
panY

EQRCodeGeometry.EQRCodeGeometry

Creates an [EQRCodeGeometry](#) object.

```
[VB6]  
void EQRCodeGeometry()  
)  
  
void EQRCodeGeometry(  
    EQRCodeGeometry other  
)  
  
void EQRCodeGeometry(  
    EQuadrilateral position,  
    ()EPoint finderPatternCenters  
)
```

Parameters

other
-
position
-
finderPatternCenters
-

EQRCodeGeometry.FinderPatternCenters

Finder patterns centers.

```
[VB6]  
FinderPatternCenters As ()EPoint  
read-only
```

Remarks

In case of a Micro QR code, there is only one finder pattern center. In case of another QR code, there are three finder pattern centers, returned in the following order: bottom left, top left, top right.

EQRCodeGeometry.operator=

```
[VB6]
EQRCodeGeometry operator=
EQRCodeGeometry other
)
```

Parameters

other

EQRCodeGeometry.Position

Position of the QR code.

```
[VB6]
Position As EQuadrilateral
read-only
```

EQRCoder Reader Class

Represents the QR code reader, that is a context for the detection and decoding of QR codes.

PROPERTIES

CellPolarityConfidenceThreshold

Sets the minimum cell polarity confidence threshold. When the cell confidence is under the threshold, additional processing is attempted to improve the polarity detection.

DetectionMethod

Sets the detection method for finding QR codes. The method can be any combination of the EQRDetectionMethod enums.

DetectionTradeOff

This setting controls the trade-off between computation speed versus reliability of the detection methods. Setting the trade-off will overwrite the current settings for EQRCoderScanPrecision and EQRDetectionMethod.

FilterOutUnreliablyDecodedQRCodes

Activate or deactivate the filtering of unreliable decoded QR codes.

ForegroundDetectionThreshold

Foreground detection threshold. This parameter determines how many grayscale-values a pixel should deviate from its local background to be considered part of the foreground.

MaximumVersion

Maximum version of QR codes to be searched for.

MinimumIsotropy	QR code minimum isotropy.
MinimumScore	Minimum pattern finder score that must be reached to consider that a finder pattern has been found.
MinimumVersion	Minimum version of QR codes to be searched for.
PerspectiveMode	Sets the perspective mode.
ScanPrecision	Precision of the QR code reader when scanning the search field.
SearchedModels	QR code models to be searched for.
SearchField	Search field for the QR code reader.
TimeOut	<p>M Time-out for the EQRCodeReader::Detect, EQRCodeReader::Decode and E EQRCodeReader::Read methods.</p>
THODS	
Decode	Decodes a QR code candidate, represented as a geometry.

Detect

Detects all QR code candidates in the search field, and returns them as a vector of geometries.

EQRCoderReader

Creates an EQRCoderReader object.

Read**EQRCoderReader.CellPolarityC**

Detects and decodes all the QR codes in the search field.

O

nfidenceThreshold

Sets the minimum cell polarity confidence threshold. When the cell confidence is under the threshold, additional processing is attempted to improve the polarity detection.

[VB6]

CellPolarityConfidenceThreshold As Single

read-write

EQRCoderReader.Decode

Decodes a QR code candidate, represented as a geometry.

[VB6]

```
EQRCode Decode(  
    EQRCodeGeometry geometries  
)
```

Parameters

geometries

Remarks

The geometry argument can either be custom-built or retrieved after a [EQRCodeReader::Detect](#).

EQRCodeReader.Detect

Detects all QR code candidates in the search field, and returns them as a vector of geometries.

[VB6]

```
() EQRCodeGeometry Detect(  
)
```

Remarks

[Detect](#) only returns candidate QR codes. These candidates can only be confirmed as actual QR codes after a successful decoding.

EQRCodeReader.DetectionMethod

Sets the detection method for finding QR codes. The method can be any combination of the EQRDetectionMethod enums.

[VB6]

DetectionMethod As Long

read-write

Remarks

The default value is: 'EQRDetectionMethod_Gradient|EQRDetectionMethod_AdaptiveThreshold'.

EQRCodeReader.DetectionTradeOff

This setting controls the trade-off between computation speed versus reliability of the detection methods. Setting the trade-off will overwrite the current settings for EQRCodeScanPrecision and EQRDetectionMethod.

[VB6]

DetectionTradeOff As EQRDetectionTradeOff

read-write

Remarks

The default value is **EQRCodeDetectionTradeOff_Balanced**.

EQRCodeReader.EQRCodeReader

Creates an EQRCodeReader object.

[VB6]

```
void EQRCodeReader(  
    )
```

EQRCodeReader.FilterOutUnreliablyDecodedQRCodes

Activate or deactivate the filtering of unreliable decoded QR codes.

[VB6]

FilterOutUnreliablyDecodedQRCodes As Boolean

read-write

Remarks

By default, the QR code reader does not return unreliable decoded QR codes.

EQRCodeReader.ForegroundDetectionThreshold

Foreground detection threshold. This parameter determines how many grayscale-values a pixel should deviate from its local background to be considered part of the foreground.

[VB6]

ForegroundDetectionThreshold As Long

read-write

Remarks

The default value for this parameter is 10.

EQRCodeReader.MaximumVersion

Maximum version of QR codes to be searched for.

[VB6]

MaximumVersion As Long

read-write

Remarks

This parameter value ranges from **1** to **40**. Default value: **40**.

EQRCoderReader.MinimumIsotropy

QR code minimum isotropy.

[VB6]

MinimumIsotropy As Single

read-write

Remarks

The isotropy of a QR code is defined as its short side divided by its long side. This parameter value ranges from 0 to 1. default value: 0.8.

EQRCoderReader.MinimumScore

Minimum pattern finder score that must be reached to consider that a finder pattern has been found.

[VB6]

MinimumScore As Single

read-write

Remarks

The pattern finder score is based on a normalized correlation with a perfect finder pattern model. A perfect match with the model would return a score of 1. This parameter value ranges from **0** to **1**. Default value: **0.65**.

EQRCoderReader.MinimumVersion

Minimum version of QR codes to be searched for.

[VB6]

MinimumVersion As Long

read-write

Remarks

This parameter value ranges from **1** to **40**. Default value: **1**.

EQRCoderReader.PerspectiveMode

Sets the perspective mode.

[VB6]

PerspectiveMode As EQRCoderPerspectiveMode

read-write

Remarks

The default value is Basic. This setting is deprecated as per Open eVision release 2.0, setting this variable will have no effect. The EQRCoderPerspectiveMode_Basic option has been superceded by EQRDetectionMethod_GradientLegacy, the EQRCoderPerspectiveMode_Improved option has been superceded by EQRDetectionMethod_PerspectiveLegacy.

EQRCoderReader.Read

Detects and decodes all the QR codes in the search field.

```
[VB6]  
() EQRCoder Read()  
 )
```

EQRCoderReader.ScanPrecision

Precision of the QR code reader when scanning the search field.

```
[VB6]  
ScanPrecision As EQRCoderScanPrecision  
read-write
```

Remarks

The default value is **EQRCoderScanPrecision_Automatic**.

EQRCoderReader.SearchedModels

QR code models to be searched for.

```
[VB6]  
SearchedModels As () EQRCoderModel  
read-write
```

Remarks

By default, the QR code reader searches for all models of QR codes.

EQRCodeReader.SearchField

Search field for the QR code reader.

[VB6]

SearchField As EROIBW8

read-write

EQRCodeReader.TimeOut

Time-out for the [EQRCodeReader::Detect](#), [EQRCodeReader::Decode](#) and [EQRCodeReader::Read](#) methods.

[VB6]

TimeOut As Unsupported variant type

read-write

Remarks

If the processing time of one of these functions becomes longer than the set time-out, the processing is stopped and an exception is thrown. In that case, the error code of the exception is [EError_TimeoutReached](#). The time-out is set in microseconds. This time-out is not a real time-out. The processing is stopped as soon as possible after the time-out has been reached. This means that the time elapsed effectively in the method can be greater than the time-out in itself.

EQuadrangle Class

This class represents a polygon with four corners (with sub-pixel accuracy).

Remarks

A quadrangle especially arises when representing the corners of a rotated bounding box.

METHODS

Draw

Draws the quadrangle, by drawing lines between its corners.

DrawWithCurrentPen

Draws the quadrangle, by drawing lines between its corners.

EQuadrangle

-

GetPoint

Returns the coordinate of a given corner of the quadrangle.

operator=

| E -

Q

uadrangle.Draw

Draws the quadrangle, by drawing lines between its corners.

[VB6]

```
void Draw(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void Draw(
    Long graphicContext,
    ERGBColor color,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void Draw(
    EDrawAdapter graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)
```

Parameters

graphicContext

Graphic context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

drawDiagonals

Specifies whether or not lines are to be drawn between the 1st and 3rd corners, as well as between the 2nd and 4th corners.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window.

EQuadrangle.DrawWithCurrentPen

Draws the quadrangle, by drawing lines between its corners.

[VB6]

```
void DrawWithCurrentPen(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)
```

Parameters

graphicContext

Graphic context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

drawDiagonals

Specifies whether or not lines are to be drawn between the 1st and 3rd corners, as well as between the 2nd and 4th corners.

Remarks

Drawing is done in the device context associated to the desired window.

EQuadrangle.EQuadrangle

-
[VB6]

```
void EQuadrangle(
    EQuadrangle other
)
void EQuadrangle(
)
```

Parameters

other
-

EQuadrangle.GetPoint

Returns the coordinate of a given corner of the quadrangle.

-
[VB6]

```
EPoint GetPoint(
    Long index
)
```

Parameters

index

The index of the corner of interest (must lie in the range between 0 and 3, inclusive).

EQuadrangle.operator=

-

[VB6]

```
EQuadrangle operator=(  
    EQuadrangle other  
)
```

Parameters

other

-

EQuadrilateral Class

Represents a quadrilateral.

PROPERTIES

Corners

M

The corners of the quadrilateral.

THODS

E

EQuadrilateral

Creates an [EQuadrilateral](#) object.

operator=

-

EQuadrilateral.Corners

The corners of the quadrilateral.

[VB6]

Corners As ()EPoint

read-only

Remarks

The corners are returned in the following order: bottom left, top left, top right, bottom right.

EQuadrilateral.EQuadrilateral

Creates an [EQuadrilateral](#) object.

[VB6]

```
void EQuadrilateral()
)
void EQuadrilateral(
    ()EPoint corners
)
void EQuadrilateral(
    EQuadrilateral other
)
```

Parameters

corners

-

other

-

EQuadrilateral.operator=

-

[VB6]

```
EQuadrilateral operator=(  
    EQuadrilateral other  
)
```

Parameters

other

-

ERectangle Class

Represents a model of a rectangle in EasyGauge.

Base Class: [EFrame](#)

PROPERTIES

SizeX

X size of the ERectangle

SizeY

M Y size of the ERectangle

E

THODS

[CopyTo](#)

	Copies all the data of the current ERectangle object into another ERectangle object, and returns it.
ERectangle	Constructs a ERectangle
GetCorners	Retrieves the coordinates of each corner of a ERectangle object.
GetEdges	Retrieves each edge of a ERectangle object.
GetMidEdges	Retrieves the center coordinates of each edge of a ERectangle object.
GetPoint	Returns the coordinates of a particular point, specified by its location in the ERectangle area.
operator=	Copies all the data from another ERectangle object into the current ERectangle object
SetFromOppositeCorners	Sets the geometric parameters (center coordinates, size, and rotation angle) of an ERectangle object.
SetFromOriginMiddleEnd	-

[SetFromThreeCorners](#)

Sets the geometric parameters (center coordinates, size, and rotation angle) of an ERectangle object.

[SetFromTwoPoints](#)

-

[SetSize](#)

Sets the size of a ERectangle object.

R

ectangle.CopyTo

Copies all the data of the current ERectangle object into another ERectangle object, and returns it.

[VB6]

```
ERectangle CopyTo(  
    ERectangle other  
)
```

Parameters

other

Pointer to the ERectangle object in which the current ERectangle object data have to be copied.

Remarks

In case of a **NULL** pointer, a new ERectangle object will be created and returned.

ERectangle.ERectangle

Constructs a ERectangle

```
[VB6]  
void ERectangle()  
)  
  
void ERectangle(  
EPoint center,  
Single sizeX,  
Single sizeY,  
Single angle  
)  
  
void ERectangle(  
EPoint origin,  
EPoint end  
)  
  
void ERectangle(  
EPoint origin,  
EPoint middle,  
EPoint end  
)  
  
void ERectangle(  
ERectangle other  
)
```

Parameters

center

Center coordinates of the rectangle at its nominal position. The default value is **(0,0)**.

sizeX

Nominal size X/Y of the rectangle. Both default values are **100**.

sizeY

Nominal size X/Y of the rectangle. Both default values are **100**.

angle

Nominal rotation angle of the rectangle. The default value is **0**.

origin

Upper left point coordinates of the rectangle.

end

Lower right point coordinates of the rectangle.

middle

A third corner point coordinates.

other

Another ERectangle object to be copied in the new ERectangle object.

ERectangle.GetCorners

Retrieves the coordinates of each corner of a ERectangle object.

[VB6]

```
void GetCorners(  
    EPoint xy,  
    EPoint XXy,  
    EPoint xYY,  
    EPoint XXYY  
)
```

Parameters

xy

Coordinates of the lower leftmost corner of the ERectangle object.

XXy

Coordinates of the lower rightmost corner of the ERectangle object.

xYY

Coordinates of the upper leftmost corner of the ERectangle object.

XXYY

Coordinates of the upper rightmost corner of the ERectangle object.

ERectangle.GetEdges

Retrieves each edge of a ERectangle object.

[VB6]

```
void GetEdges(
    ELine x,
    ELine XX,
    ELine y,
    ELine YY
)
```

Parameters

x

Leftmost edge of the ERectangle object.

XX

Rightmost edge of the ERectangle object.

y

Lower edge of the ERectangle object.

YY

Upper edge of the ERectangle object.

ERectangle.GetMidEdges

Retrieves the center coordinates of each edge of a ERectangle object.

[VB6]

```
void GetMidEdges(
    EPoint x,
    EPoint XX,
    EPoint y,
    EPoint YY
)
```

Parameters

x

Center coordinates of the leftmost edge of the ERectangle object.

XX

Center coordinates of the rightmost edge of the ERectangle object.

y

Center coordinates of the lower edge of the ERectangle object.

YY

Center coordinates of the upper edge of the ERectangle object.

ERectangle.GetPoint

Returns the coordinates of a particular point, specified by its location in the ERectangle area.

[VB6]

```
EPoint GetPoint(
    Single fractionX,
    Single fractionY
)
```

Parameters

fractionX

Point location expressed as a fraction of the ERectangle vertical edges (range -1, +1).

fractionY

Point location expressed as a fraction of the ERectangle horizontal edges (range -1, +1).

ERectangle.operator=

Copies all the data from another ERectangle object into the current ERectangle object

[VB6]

```
ERectangle operator=(
    ERectangle other
)
```

Parameters

other

ERectangle object to be copied

ERectangle.SetFromOppositeCorners

Sets the geometric parameters (center coordinates, size, and rotation angle) of an ERectangle object.

[VB6]

```
void SetFromOppositeCorners(
    EPoint origin,
    EPoint end
)
```

Parameters

origin

Upper left point coordinates of the rectangle.

end

Lower right point coordinates of the rectangle.

ERectangle.SetFromOriginMiddleEnd

[VB6]

```
void SetFromOriginMiddleEnd(
    EPoint origin,
    EPoint middle,
    EPoint end
)
```

Parameters

origin

-

middle

-

end

ERectangle.SetFromThreeCorners

Sets the geometric parameters (center coordinates, size, and rotation angle) of an ERectangle object.

[VB6]

```
void SetFromThreeCorners(  
    EPoint origin,  
    EPoint middle,  
    EPoint end  
)
```

Parameters

origin

Upper left point coordinates of the rectangle.

middle

A third corner point coordinates.

end

Lower right point coordinates of the rectangle.

ERectangle.SetFromTwoPoints

[VB6]

```
void SetFromTwoPoints(  
    EPoint origin,  
    EPoint end  
)
```

Parameters

origin

-

end

-

ERectangle.SetSize

Sets the size of a ERectangle object.

[VB6]

```
void SetSize(  
    Single sizeX,  
    Single sizeY  
)
```

Parameters

sizeX

Nominal size X of the ERectangle object. Default values is **100**.

sizeY

Nominal size Y of the ERectangle object. Default values is **100**.

Remarks

A ERectangle object is fully defined knowing its nominal position (given by the coordinates of its center), its nominal size, its rotation angle and its outline tolerance. By default, the width and height values are **100**, which means 100 pixels when the field of view is not calibrated and 100 "units" in case of a calibrated field of view.

ERectangle.SizeX

X size of the ERectangle

[VB6]

SizeX As Single

read-only

ERectangle.SizeY

Y size of the ERectangle

[VB6]

SizeY As Single

read-only

ERectangleGauge Class

Manages a rectangle fitting gauge.

Base Class: [ERectangleShape](#)

PROPERTIES

[Active](#)

Sets the flag indicating whether the gauge is active or not.

[ActiveEdges](#)

Active edges as defined in [EDragHandle](#).

AverageDistance	Average distance between the sampled points and the fitted model.
FilteringThreshold	Relative filtering threshold, that is the fraction of the average distance between the sampled points and the fitted model above which a point is filtered out.
HVConstraint	-
InnerFilteringEnabled	Getter method for the GetInnerFilteringEnabled property. This property is the flag indicating if the inner sampled point filtering is enabled (TRUE).
InnerFilteringThreshold	Sampled point inner filtering threshold.
KnownAngle	Flag indicating whether the rotation angle of the rectangle to be fitted is known or not.
MeasuredRectangle	Information pertaining to the fitted rectangle.
MinAmplitude	-
MinArea	-
NumFilteringPasses	Number of filtering passes for a model fitting operation.

NumSamples	Number of sampled points during the model fitting operation.
NumSamplesx	Number of sampled points found on edge x during the measure operation.
NumSamplesX	Number of sampled points found on edge X during the measure operation.
NumSamplesy	Number of sampled points found on edge y during the measure operation.
NumSamplesY	Number of sampled points found on edge Y during the measure operation.
NumSkipRanges	Number of skip ranges in the gauge after a call to ERectangleGauge::AddSkipRange .
NumValidSamples	Number of valid sample points remaining after a model fitting operation.
RectangularSamplingArea	-
SamplingStep	Approximate distance between sampled points during a model fitting operation.
Shape	-

Smoothing	-
Thickness	-
Threshold	-
Tolerance	Searching area half thickness of the rectangle fitting gauge.
TransitionChoice	-
TransitionIndex	-
TransitionType	-
Type	Shape type.
Valid	M E
THODS	
AddSkipRange	Adds an item to the set of skip ranges and returns the index of the newly added range.

[CopyTo](#)

Copies all the data of the current ERectangleGauge object into another ERectangleGauge object, and returns it.

[DisableInnerFiltering](#)

Disables inner sampled point filtering.

[Drag](#)

Moves a handle to a new position and updates the position parameters of the gauge.

[Draw](#)

Draws a graphical representation of a point location or model fitting gauge, as defined by [EDrawingMode](#).

[DrawWithCurrentPen](#)

Draws a graphical representation of a point location or model fitting gauge, as defined by [EDrawingMode](#).

[ERectangleGauge](#)

Constructs a rectangle measurement context.

[GetMeasuredPoint](#)

Returns the coordinates of a sample point, measured along one of the sample paths of the gauge.

[GetMinNumFitSamples](#)

Returns the minimum number of samples required for fitting on each side of the shape.

GetSamplex	Allows to retrieve information on the samples found along the x edge.
GetSampleX	Allows to retrieve information on the samples found along the X edge.
GetSampley	Allows to retrieve information on the samples found along the y edge.
GetSampleY	Allows to retrieve information on the samples found along the Y edge.
GetSkipRange	Allows to retrieve the start and end values of the skip range corresponding to the given index, if it is valid (i.e. if it has previously been created by a call to the ERectangleGauge::AddSkipRange method).
HitTest	Checks whether the cursor is positioned over a handle (TRUE) or not (FALSE).
Measure	Triggers the point location or the model fitting operation.
MeasureSample	Computes the sample points along the sample path whose index in the list is given by the pathIndex parameter.

MeasureWithoutFitting

Triggers the point location without rectangle fitting operation.

operator=

Copies all the data from another ERectangleGauge object into the current ERectangleGauge object

Plot

Draws the profile that is crossed by one of the sample paths of the gauge, as defined by [EPlotItem](#).

PlotWithCurrentPen

Draws the profile that is crossed by one of the sample paths of the gauge, as defined by [EPlotItem](#).

Process

Triggers the process pertaining to a shape or gauge and all the daughter gauges attach to it.

RemoveAllSkipRanges

Removes all the skip ranges previously created by a call to [ERectangleGauge::AddSkipRange](#).

RemoveSkipRange

After a call to [ERectangleGauge::AddSkipRange](#), removes the skip range with the given index.

SetMinNumFitSamples

Sets the minimum number of samples required for fitting on each side of the shape.

ERectangleGauge.Active

Sets the flag indicating whether the gauge is active or not.

[VB6]

Active As Boolean

read-write

Remarks

When complex gauging is required, several gauges can be grouped together. Applying [ERectangleGauge::Process](#) to the mother gauge or shape triggers the measurement of the whole. Only the active gauges will participate in the process. By default, the gauge is active (**TRUE**).

ERectangleGauge.ActiveEdges

Active edges as defined in [EDragHandle](#).

[VB6]

ActiveEdges As Long

read-write

Remarks

In the case of a rectangle fitting gauge, each edge can have its own transition detection parameters. Updating the transition parameters only affect the current active edges. By default, all edges are active.

ERectangleGauge.AddSkipRange

Adds an item to the set of skip ranges and returns the index of the newly added range.

[VB6]

```
Long AddSkipRange(
    Long start,
    Long end
)
```

Parameters

start

Beginning of the skip range.

end

End of the skip range.

Remarks

The samples indices between start and end (including the boundaries) will be discarded during the measurement process. The [AddSkipRange](#) method allows to define skip ranges in an [ERectangleGauge](#). This means that, at measure time, samples belonging to these ranges will not be taken into account. A sample may belong to more than one skip range; to be discarded, a sample has to pertain to at least one range. Moreover, the skip ranges are allowed to overlap one another. The range is allowed to be reversed (i.e. end is not required to be greater than start). Also, start and end are not required to reference valid indices at the time of the call (i.e. the range may lie outside of the current return value for [ERectangleGauge::NumSamples](#)).

ERectangleGauge.AverageDistance

Average distance between the sampled points and the fitted model.

[VB6]

AverageDistance As Single

read-only

Remarks

Irrelevant in case of a point location operation.

ERectangleGauge.CopyTo

Copies all the data of the current ERectangleGauge object into another ERectangleGauge object, and returns it.

[VB6]

```
ERectangleGauge CopyTo(
    ERectangleGauge other,
    Boolean recursive
)
```

Parameters

other

Pointer to the ERectangleGauge object in which the current ERectangleGauge object data have to be copied.

recursive

TRUE if the children gauges have to be copied as well, **FALSE** otherwise.

Remarks

In case of a **NULL** pointer, a new ERectangleGauge object will be created and returned.

ERectangleGauge.DisableInnnerFiltering

Disables inner sampled point filtering.

[VB6]

```
void DisableInnnerFiltering(
)
```

Remarks

The inner sampled point filtering is activated as soon as the corresponding [ERectangleGauge::InnerFilteringThreshold](#) is set.

ERectangleGauge.Drag

Moves a handle to a new position and updates the position parameters of the gauge.

```
[VB6]  
void Drag(  
    Long x,  
    Long y  
)
```

Parameters

x
Cursor current coordinates.
y
Cursor current coordinates.

ERectangleGauge.Draw

Draws a graphical representation of a point location or model fitting gauge, as defined by [EDrawingMode](#).

```
[VB6]  
void Draw(  
    EDrawAdapter graphicContext,  
    EDrawingMode drawingMode,  
    Boolean daughters  
)  
  
void Draw(  
    Long graphicContext,  
    EDrawingMode drawingMode,  
    Boolean daughters  
)
```

```
void Draw(
    Long graphicContext,
    ERGBColor color,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

Parameters*graphicContext*

Handle of the device context on which to draw.

*drawingMode*Indicates how the point location or model fitting gauge must be displayed, as defined by [EDrawingMode](#).*daughters***TRUE** if the daughters gauges are to be displayed also.*color*

The color in which to draw the overlay.

ERectangleGauge.DrawWithCurrentPen

Draws a graphical representation of a point location or model fitting gauge, as defined by [EDrawingMode](#).

[VB6]

```
void DrawWithCurrentPen(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

Parameters*graphicContext*

Handle of the device context on which to draw.

*drawingMode*Indicates how the point location or model fitting gauge must be displayed, as defined by [EDrawingMode](#).

*daughters***TRUE** if the daughters gauges are to be displayed also.

ERectangleGauge.ERectangleGauge

Constructs a rectangle measurement context.

[VB6]

```
void ERectangleGauge()
)

void ERectangleGauge(
    ERectangleGauge other
)
```

Parameters

other

Another ERectangleGauge object to be copied in the new ERectangleGauge object.

Remarks

With the default constructor, all the parameters are initialized to their respective default values. With the copy constructor, the constructed rectangle measurement context is based on a pre-existing ERectangleGauge object. The gauge children are also copied. Hierarchy copying through a copy constructor is always recursive. To disable this recursion, use instead the [ERectangleGauge::CopyTo](#) method.

ERectangleGauge.FilteringThreshold

Relative filtering threshold, that is the fraction of the average distance between the sampled points and the fitted model above which a point is filtered out.

[VB6]

```
FilteringThreshold As Single
```

read-write

Remarks

Irrelevant in case of a point location operation. During a model fitting operation, the "filtering" process can be invoked to remove outliers, i.e. points that were located significantly far away from the fitted model (so that their position is dubious).

ERectangleGauge.GetMeasuredPoint

Returns the coordinates of a sample point, measured along one of the sample paths of the gauge.

[VB6]

```
EPoint GetMeasuredPoint(
    Long index
)
```

Parameters*index*

This argument must be left unchanged from its default value, i.e. **~0 (= 0xFFFFFFFF)**.

Remarks

These coordinates pertain to the world space; they are expressed in the reference frame to which the current ERectangleGauge object belongs. The gauging process uses a list of sample points to find the shape position and size that best fit a given image. These sample points are measured along the sample paths defined by the gauge geometry.

[ERectangleGauge::GetMeasuredPoint](#) returns the coordinates of the sample point that meets the following two requirements: 1. It lies on the sample path inspected with the last call to [ERectangleGauge::MeasureSample](#), and 1. Among all the sample points along the latter sample path, it is the one selected by the [ERectangleGauge::TransitionChoice](#) property.

Note. For this method to succeed, it is necessary to previously call [ERectangleGauge::MeasureSample](#).

ERectangleGauge.GetMinNumFitSamples

Returns the minimum number of samples required for fitting on each side of the shape.

[VB6]

```
void GetMinNumFitSamples(  
    Long side0,  
    Long side1,  
    Long side2,  
    Long side3  
)
```

Parameters

side0

Minimum number of samples on the top side of the rectangle.

side1

Minimum number of samples on the left side of the rectangle.

side2

Minimum number of samples on the bottom side of the rectangle.

side3

Minimum number of samples on the right side of the rectangle.

Remarks

Irrelevant in case of a point location operation.

ERectangleGauge.GetSampleX

Allows to retrieve information on the samples found along the X edge.

[VB6]

```

Boolean GetSampleX(
    EPoint point,
    Long index
)

Boolean GetSampleX(
    EPeak peak,
    Long index
)

```

Parameters

point
-
index
 The sample index
peak
-

Remarks

The method provides the sample point corresponding to the supplied index. The returned value is true when the sample is valid, and false otherwise.

ERectangleGauge.GetSampleX

Allows to retrieve information on the samples found along the X edge.

[VB6]

```

Boolean GetSampleX(
    EPoint point,
    Long index
)

Boolean GetSampleX(
    EPeak peak,
    Long index
)

```

Parameters

point

-
index
 The sample index
peak
-

Remarks

The method provides the sample point corresponding to the supplied index. The returned value is true when the sample is valid, and false otherwise.

ERectangleGauge.GetSampleY

Allows to retrieve information on the samples found along the Y edge.

```
[VB6]  
  
Boolean GetSampleY(  
    EPoint pt,  
    Long index  
)  
  
Boolean GetSampleY(  
    EPeak pk,  
    Long index  
)
```

Parameters

pt
 EPoint structure that will receive the sample position.
index
 The sample index
pk
-

Remarks

The method provides the sample point corresponding to the supplied index. The returned value is true when the sample is valid, and false otherwise.

ERectangleGauge.GetSampleY

Allows to retrieve information on the samples found along the Y edge.

```
[VB6]

Boolean GetSampleY(
    EPoint pt,
    Long index
)

Boolean GetSampleY(
    EPeak pk,
    Long index
)
```

Parameters

pt

[EPoint](#) structure that will receive the sample position.

index

The sample index

pk

-

Remarks

The method provides the sample point corresponding to the supplied index. The returned value is true when the sample is valid, and false otherwise.

ERectangleGauge.GetSkipRange

Allows to retrieve the start and end values of the skip range corresponding to the given index, if it is valid (i.e. if it has previously been created by a call to the [ERectangleGauge::AddSkipRange](#) method).

```
[VB6]
```

```
void GetSkipRange(
    Long index,
    Long start,
    Long end
)
```

Parameters

index

Index of the skip range.

start

Beginning of the skip range.

end

End of the skip range.

Remarks

Start is guaranteed to be smaller or equal to end.

ERectangleGauge.HitTest

Checks whether the cursor is positioned over a handle (**TRUE**) or not (**FALSE**).

[VB6]

```
Boolean HitTest(
    Boolean daughters
)
```

Parameters

daughters

TRUE if the daughters gauges handles have to be considered as well.

ERectangleGauge.HVConstraint

[VB6]

HVConstraint As Boolean

read-write

ERectangleGauge.InnerFilteringEnabled

Getter method for the **GetInnerFilteringEnabled** property. This property is the flag indicating if the inner sampled point filtering is enabled (**TRUE**).

[VB6]

InnerFilteringEnabled As Boolean

read-only

Remarks

The inner sampled point filtering is activated as soon as the corresponding threshold is set, getting the [ERectangleGauge.InnerFilteringThreshold](#) property. To disable inner filtering, use the **DisableInnerFiltering** method.

ERectangleGauge.InnerFilteringThreshold

Sampled point inner filtering threshold.

[VB6]

InnerFilteringThreshold As Single

read-write

Remarks

If inner filtering is enabled, the sampled points that have been found inside the measured rectangle are filtered in regard of their distance to it. If this distance is greater than the threshold, the sampled point is set as invalid, and removed from the measure. This distance is in physical units. The inner sampled point filtering is activated as soon as the corresponding threshold is set. To disable inner filtering, use the **DisableInnerFiltering** method.

ERectangleGauge.KnownAngle

Flag indicating whether the rotation angle of the rectangle to be fitted is known or not.

[VB6]

KnownAngle As Boolean

read-write

Remarks

A rectangle model to be fitted may have a well-known orientation. It is possible to impose the value of this rotation angle, thus removing one degree of freedom. The rectangle fitting gauge orientation is set by means of the [ERectangleGauge.Angle](#) property. The sign of the rotation angle depends whether the field of view is calibrated or not. * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value. * When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

ERectangleGauge.Measure

Triggers the point location or the model fitting operation.

[VB6]

```
void Measure(
    EROIBW8 sourceImage
)
```

Parameters

sourceImage

Pointer to the source image.

Remarks

When this method is called, and if not enough valid points are detected, then the method returns directly, and the measured gauge is set to the nominal parameters.

ERectangleGauge.MeasuredRectangle

Information pertaining to the fitted rectangle.

[VB6]

MeasuredRectangle As ERectangle

read-only

ERectangleGauge.MeasureSample

Computes the sample points along the sample path whose index in the list is given by the **pathIndex** parameter.

[VB6]

```
void MeasureSample(
    EROIBW8 sourceImage,
    Long pathIndex
)
```

Parameters*sourceImage*

Pointer to the source image/ROI.

pathIndex

Sample path index.

Remarks

This method stores its results into a temporary variable inside the ERectangleGauge object.

ERectangleGauge.MeasureWithoutFitting

Triggers the point location without rectangle fitting operation.

[VB6]

```
void MeasureWithoutFitting(
    EROIBW8 sourceImage
)
```

Parameters*sourceImage*

Source image.

Remarks

This method performs the actual measurement for each transition, but does not perform the rectangle fitting. This means that individual samples will be available for each edges through the [ERectangleGauge::GetSamplex](#) (Edge x), [ERectangleGauge::GetSampley](#) (Edge y), [ERectangleGauge::GetSampleX](#) (Edge X), [ERectangleGauge::GetSampleY](#) (Edge Y) methods, but the gauge position will not be changed. Please note that the filtering will not be performed in this method, since it relies upon the fitting process. The filtering parameters will thus be unused.

ERectangleGauge.MinAmplitude

[VB6]

MinAmplitude As Long

read-write

ERectangleGauge.MinArea

-

[VB6]

MinArea As Long

read-write

ERectangleGauge.NumFilteringPasses

Number of filtering passes for a model fitting operation.

[VB6]

NumFilteringPasses As Long

read-write

Remarks

Irrelevant in case of a point location operation. During a filtering pass, the points that are too distant from the model are discarded. During a model fitting operation, the "filtering" process can be invoked to remove outliers, i.e. points that were located significantly far away from the fitted model (so that their position is dubious). By default (the number of filtering passes is 0), the outliers rejection process is disabled.

ERectangleGauge.NumSamples

Number of sampled points during the model fitting operation.

[VB6]

NumSamples As Long

read-only

Remarks

Irrelevant in case of a point location operation. After a model fitting operation, a number of points have been fitted along the model. Among them, some are not reliable because of their **Area** value. Among the remaining ones, some are filtered out (**NumFilteringPasses**, **FilteringThreshold**).

ERectangleGauge.NumSamplesX

Number of sampled points found on edge X during the measure operation.

[VB6]

NumSamplesX As Long

read-only

ERectangleGauge.NumSamplesX

Number of sampled points found on edge X during the measure operation.

[VB6]

NumSamplesX As Long

read-only

ERectangleGauge.NumSamplesY

Number of sampled points found on edge Y during the measure operation.

[VB6]

NumSamplesY As Long

read-only

ERectangleGauge.NumSamplesY

Number of sampled points found on edge Y during the measure operation.

[VB6]

NumSamplesY As Long

read-only

ERectangleGauge.NumSkipRanges

Number of skip ranges in the gauge after a call to [ERectangleGauge::AddSkipRange](#).

[VB6]

NumSkipRanges As Long

read-only

ERectangleGauge.NumValidSamples

Number of valid sample points remaining after a model fitting operation.

[VB6]

NumValidSamples As Long

read-only

Remarks

Irrelevant in case of a point location operation. After a model fitting operation, a number of points have been fitted along the model. Among them, some are not reliable because of their **Area** value. Among the remaining ones, some are filtered out (**NumFilteringPasses**, **FilteringThreshold**).

ERectangleGauge.operator=

Copies all the data from another ERectangleGauge object into the current ERectangleGauge object

[VB6]

```
ERectangleGauge operator=(
    ERectangleGauge other
)
```

Parameters

other

ERectangleGauge object to be copied

ERectangleGauge.Plot

Draws the profile that is crossed by one of the sample paths of the gauge, as defined by [EPlotItem](#).

[VB6]

```
void Plot(
    EDrawAdapter graphicContext,
    EPlotItem drawItems,
    Single width,
    Single height,
    Single originX,
    Single originY
)

void Plot(
    Long graphicContext,
    EPlotItem drawItems,
    Single width,
    Single height,
    Single originX,
    Single originY
)

void Plot(
    Long graphicContext,
    ERGBColor color,
    EPlotItem drawItems,
    Single width,
    Single height,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawItems

Boolean combination of [EPlotItem](#) members, that indicates which items must be displayed.

width

Width of the plot.

height

Height of the plot.

originX

Origin point coordinates of the plot along the X axis.

originY

Origin point coordinates of the plot along the Y axis.

color

The color in which to draw the overlay.

Remarks

The sample path that is taken into considered is the one inspected with the last call to [ERectangleGauge::MeasureSample](#).

Note. For this method to succeed, it is necessary to previously call [ERectangleGauge::MeasureSample](#).

ERectangleGauge.PlotWithCurrentPen

Draws the profile that is crossed by one of the sample paths of the gauge, as defined by [EPlotItem](#).

[VB6]

```
void PlotWithCurrentPen(
    Long graphicContext,
    EPlotItem drawItems,
    Single width,
    Single height,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

*drawItems*Boolean combination of [EPlotItem](#) members, that indicates which items must be displayed.*width*

Width of the plot.

height

Height of the plot.

originX

Origin point coordinates of the plot along the X axis.

originY

Origin point coordinates of the plot along the Y axis.

Remarks

The sample path that is taken into considered is the one inspected with the last call to [ERectangleGauge::MeasureSample](#).

Note. For this method to succeed, it is necessary to previously call [ERectangleGauge::MeasureSample](#).

ERectangleGauge.Process

Triggers the process pertaining to a shape or gauge and all the daughter gauges attach to it.

[VB6]

```
void Process(
    EROIBW8 sourceImage,
    Boolean daughters
)
```

Parameters

sourceImage

Pointer to the source image.

daughters

Flag indicating whether the daughters shapes inherit of the same behavior.

Remarks

When complex gauging is required, several gauges can be grouped together. Applying **Process** to the mother gauge or shape triggers the measurement of the whole. Only the active gauges will participate in the process.

ERectangleGauge.RectangularSamplingArea

-

[VB6]

RectangularSamplingArea As Boolean

read-write

ERectangleGauge.RemoveAllSkipRanges

Removes all the skip ranges previously created by a call to
[ERectangleGauge::AddSkipRange](#).

[VB6]

```
void RemoveAllSkipRanges()  
 )
```

ERectangleGauge.RemoveSkipRange

After a call to [ERectangleGauge::AddSkipRange](#), removes the skip range with the given index.

[VB6]

```
void RemoveSkipRange(  
 Long index  
 )
```

Parameters

index

Index of the skip range to remove, as returned by [ERectangleGauge::AddSkipRange](#).

ERectangleGauge.SamplingStep

Approximate distance between sampled points during a model fitting operation.

[VB6]

SamplingStep As Single

read-write

Remarks

Irrelevant in case of a point location operation. To fit a model, a series of point location operations are performed along the model. The point location gauges spacing is given by the sampling step. By default, the sampling step is set to **5**, which means 5 pixels when the field of view is not calibrated, and 5 physical units in case of a calibrated field of view. Be aware that if the sampling step is too large, the number of sampled points along the model will not be sufficient enough to accurately locate it.

ERectangleGauge.SetMinNumFitSamples

Sets the minimum number of samples required for fitting on each side of the shape.

[VB6]

```
void SetMinNumFitSamples(  
    Long side0,  
    Long side1,  
    Long side2,  
    Long side3  
)
```

Parameters

`side0`

Minimum number of samples on the *top side* of the rectangle. The default value is **2**.

`side1`

Minimum number of samples on the *left side* of the rectangle. If this value is not specified, it is equal to **n32Side0**. The default value is **2**.

`side2`

Minimum number of samples on the *bottom side* of the rectangle. If this value is not specified, it is equal to **n32Side0**. The default value is **2**.

`side3`

Minimum number of samples on the *right side* of the rectangle. If this value is not specified, it is equal to **n32Side1**. The default value is **2**.

Remarks

When the **ERectangleGauge::Measure** method is called, and if not enough valid points are detected, then the method returns directly, and the measured gauge is set to the nominal parameters.

ERectangleGauge.Shape

-

[VB6]

Shape As ERectangle

read-only

ERectangleGauge.Smoothing

-

[VB6]

Smoothing As Long

read-write

ERectangleGauge.Thickness

-

[VB6]

Thickness As Long

read-write

ERectangleGauge.Threshold

-

[VB6]

Threshold As Long

read-write

ERectangleGauge.Tolerance

Searching area half thickness of the rectangle fitting gauge.

[VB6]

Tolerance As Single

read-write

Remarks

A rectangle fitting gauge is fully defined knowing its nominal position (its center coordinates), its nominal size, its rotation angle, and its outline tolerance. By default, the searching area thickness of the rectangle fitting gauge is **20** (2x10), which means 20 pixels when the field of view is not calibrated, and 20 physical units in case of a calibrated field of view.

ERectangleGauge.TransitionChoice

-

[VB6]

TransitionChoice As ETransitionChoice

read-write

ERectangleGauge.TransitionIndex

-

[VB6]

TransitionIndex As Long

read-write

ERectangleGauge.TransitionType

-

[VB6]

TransitionType As ETransitionType

read-write

ERectangleGauge.Type

Shape type.

[VB6]

Type As EShapeType

read-only

ERectangleGauge.Valid

-

[VB6]

Valid As Boolean

read-only

ERectangleShape Class

-

Base Class: [EShape](#)

Derived Class(es): [EBarcode](#) [ERectangleGauge](#)

PROPERTIES

Angle	-
Center	-
CenterX	-
CenterY	-
Rectangle	-
Scale	-
SizeX	X size of the ERectangleShape
SizeY	Y size of the ERectangleShape
Type	M Shape type. E

THODS

Closest	Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use EShape::ClosestShape .
---------	--

CopyTo	-
Drag	-
Draw	Draws a graphical representation of a shape, as defined by EDrawingMode .
DrawWithCurrentPen	Draws a graphical representation of a shape, as defined by EDrawingMode .
GetCorners	Retrieves the coordinates of each corner of a ERectangleShape object.
GetEdges	Retrieves each edge of a ERectangleShape object.
GetMidEdges	Retrieves the center coordinates of each edge of a ERectangleShape object.
GetPoint	Returns the coordinates of a particular point, specified by its location in the ERectangleShape area.
HitTest	-
operator=	Copies all the data from another ERectangleShape object into the current ERectangleShape object

SetCenterXY	Sets the center coordinates of a ERectangleShape object.
SetFromOppositeCorners	-
SetFromOriginMiddleEnd	-
SetFromThreeCorners	-
SetFromTwoPoints	-
SetSize	Sets the size of a ERectangleShape object. E
R	

ectangleShape.Angle

-

Angle As Single

read-write

ERectangleShape.Center

-

[VB6]

Center As EPoint

read-write

ERectangleShape.CenterX

-

[VB6]

CenterX As Single

read-only

ERectangleShape.CenterY

-

[VB6]

CenterY As Single

read-only

ERectangleShape.Closest

Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use [EShape::ClosestShape](#).

[VB6]

```
void Closest(  
)
```

ERectangleShape.CopyTo

-

[VB6]

```
ERectangleShape CopyTo(  
    ERectangleShape dest,  
    Boolean bRecursive  
)
```

Parameters

dest

bRecursive

ERectangleShape.Drag

-

[VB6]

```
void Drag(  
    Long n32CursorX,  
    Long n32CursorY  
)
```

Parameters

n32CursorX

-

n32CursorY

-

ERectangleShape.Draw

Draws a graphical representation of a shape, as defined by [EDrawingMode](#).

[VB6]

```
void Draw(
    EDrawAdapter graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)

void Draw(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)

void Draw(
    Long graphicContext,
    ERGBColor color,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by [EDrawingMode](#).

daughters

TRUE if the daughters gauges are to be displayed also.

color

-

ERectangleShape.DrawWithCurrentPen

Draws a graphical representation of a shape, as defined by [EDrawingMode](#).

[VB6]

```
void DrawWithCurrentPen(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by [EDrawingMode](#).

daughters

TRUE if the daughters gauges are to be displayed also.

ERectangleShape.GetCorners

Retrieves the coordinates of each corner of a ERectangleShape object.

[VB6]

```
void GetCorners(
    EPoint xy,
    EPoint XXy,
    EPoint xYY,
    EPoint XXYY
)
```

Parameters

xy

Coordinates of the lower leftmost corner of the ERectangleShape object.

XXY

Coordinates of the lower rightmost corner of the ERectangleShape object.

XY

Coordinates of the upper leftmost corner of the ERectangleShape object.

XXYY

Coordinates of the upper rightmost corner of the ERectangleShape object.

ERectangleShape.GetEdges

Retrieves each edge of a ERectangleShape object.

[VB6]

```
void GetEdges(
    ELine x,
    ELine XX,
    ELine y,
    ELine YY
)
```

Parameters

x

Leftmost edge of the ERectangleShape object.

XX

Rightmost edge of the ERectangleShape object.

y

Lower edge of the ERectangleShape object.

YY

Upper edge of the ERectangleShape object.

ERectangleShape.GetMidEdges

Retrieves the center coordinates of each edge of a ERectangleShape object.

[VB6]

```
void GetMidEdges(
    EPoint x,
    EPoint XX,
    EPoint y,
    EPoint YY
)
```

Parameters

x

Center coordinates of the leftmost edge of the ERectangleShape object.

XX

Center coordinates of the rightmost edge of the ERectangleShape object.

y

Center coordinates of the lower edge of the ERectangleShape object.

YY

Center coordinates of the upper edge of the ERectangleShape object.

ERectangleShape.GetPoint

Returns the coordinates of a particular point, specified by its location in the ERectangleShape area.

[VB6]

```
EPoint GetPoint(
    Single fractionX,
    Single fractionY
)
```

Parameters

fractionX

Point location expressed as a fraction of the ERectangleShape vertical edges (range -1, +1).

fractionY

Point location expressed as a fraction of the ERectangleShape horizontal edges (range -1, +1).

ERectangleShape.HitTest

-

[VB6]

```
Boolean HitTest(  
    Boolean bDaughters  
)
```

Parameters

bDaughters

ERectangleShape.operator=

Copies all the data from another ERectangleShape object into the current ERectangleShape object

[VB6]

```
ERectangleShape operator=(  
    ERectangleShape other  
)
```

Parameters

other

ERectangleShape object to be copied

ERectangleShape.Rectangle

-

[VB6]

Rectangle As ERectangle

read-write

ERectangleShape.Scale

-

[VB6]

Scale As Single

read-write

ERectangleShape.SetCenterXY

Sets the center coordinates of a ERectangleShape object.

[VB6]

```
void SetCenterXY(  
    Single centerX,  
    Single centerY  
)
```

Parameters

centerX

Center coordinates of the ERectangleShape object.

centerY

Center coordinates of the ERectangleShape object.

ERectangleShape.SetFromOppositeCorners

-

[VB6]

```
void SetFromOppositeCorners (
    EPoint origin,
    EPoint end
)
```

Parameters

origin

end

ERectangleShape.SetFromOriginMiddleEnd

-

[VB6]

```
void SetFromOriginMiddleEnd (
    EPoint origin,
    EPoint middle,
    EPoint end
)
```

Parameters

origin

middle

end

ERectangleShape.SetFromThreeCorners

[VB6]

```
void SetFromThreeCorners(
    EPoint origin,
    EPoint middle,
    EPoint end
)
```

Parameters

origin

middle

end

ERectangleShape.SetFromTwoPoints

[VB6]

```
void SetFromTwoPoints(
    EPoint origin,
    EPoint end
)
```

Parameters

origin

-
end
-

ERectangleShape.SetSize

Sets the size of a ERectangleShape object.

[VB6]

```
void SetSize(  
    Single sizeX,  
    Single sizeY  
)
```

Parameters

sizeX

Nominal size X of the ERectangleShape object. Default values is **100**.

sizeY

Nominal size Y of the ERectangleShape object. Default values is **100**.

Remarks

A ERectangleShape object is fully defined knowing its nominal position (given by the coordinates of its center), its nominal size, its rotation angle and its outline tolerance. By default, the width and height values are **100**, which means 100 pixels when the field of view is not calibrated and 100 "units" in case of a calibrated field of view.

ERectangleShape.SizeX

X size of the ERectangleShape

[VB6]

```
SizeX As Single
```

read-only

ERectangleShape.SizeY

Y size of the ERectangleShape

[VB6]

SizeY As Single

read-only

ERectangleShape.Type

Shape type.

[VB6]

Type As EShapeType

read-only

EReferenceImageSegmenter Class

Segments an image using a pixel-by-pixel single threshold given as an image.

Remarks

This segmenter is applicable to [EROIBW8](#), [EROIBW16](#) and [EROIC24](#) images. It produces coded images with two layers. The threshold is defined for each pixel individually by means of a reference image of the same type as the source image.

For grayscales images, the White layer (usually, with index 1) contains unmasked pixels having a gray value in a range defined by the gray value of the respective pixel in the reference image and the white color.

For RGB color images, the White layer (usually, with index 1) contains unmasked pixels having a color inside the cube of the RGB color space defined by the color of the respective pixel in the reference image and the white color (255,255,255).

The Black layer (usually, with index 0) contains the remaining unmasked pixels.

Base Class: [ETwoLayersImageSegmenter](#)

PROPERTIES

BlackLayerEncoded	Black layer encoding status.
BlackLayerIndex	Index of the black layer in the destination coded image.
ReferenceImageBW16	Reference image for the segmentation of EROIBW16 images.
ReferenceImageBW8	Reference image for the segmentation of EROIBW8 images.
ReferenceImageC24	Reference image for the segmentation of EROIC24 images.
WhiteLayerEncoded	White layer encoding status.

WhiteLayerIndex

E Index of the white layer in the destination coded image.
R

EReferecelmageSegmenter.BlackLayerEncoded

Black layer encoding status.

[VB6]

BlackLayerEncoded As Boolean

read-write

EReferecelmageSegmenter.BlackLayerIndex

Index of the black layer in the destination coded image.

[VB6]

BlackLayerIndex As Long

read-write

Remarks

Setting this property automatically switches on the encoding of the black layer.

EReferecelmageSegmenter.ReferenceImageBW16

Reference image for the segmentation of [EROIBW16](#) images.

[VB6]

ReferenceImageBW16 As EROIBW16

read-write

EReferecelmageSegmenter.ReferenceImageBW8

Reference image for the segmentation of [EROIBW8](#) images.

[VB6]

ReferenceImageBW8 As EROIBW8

read-write

EReferecelmageSegmenter.ReferenceImageC24

Reference image for the segmentation of [EROIC24](#) images.

[VB6]

ReferenceImageC24 As EROIC24

read-write

EReferecelmageSegmenter.WhiteLayerEncoded

White layer encoding status.

[VB6]

WhiteLayerEncoded As Boolean

read-write

ERefERENCEImageSegmenter.WhiteLayerIndex

Index of the white layer in the destination coded image.

[VB6]

WhiteLayerIndex As Long

read-write

Remarks

Setting this property automatically switches on the encoding of the white layer.

EROI Class

Represents a ROI.

Base Class: [EPixelRegion](#)

Derived Class(es): [EDepthMapROI](#)

PROPERTIES

Children

Children ROIs.

Height

ROI height.

IsVoid	Tests whether if the EROI .GetRootROI@ of this hierarchy has a zero size.
OrgX	Abscissa of the ROI origin in its parent.
OrgY	Ordinate of the ROI origin in its parent.
Parent	Parent ROI.
PixelContainer	Attached pixel container.
RowPitch	Buffer row pitch.
TotalOrgX	Abscissa of the ROI origin in the root ROI.
TotalOrgY	Ordinate of the ROI origin in the root ROI.
Width	M ROI width. E
THODS	
AsEBaseROI	-
Attach	Attaches the ROI to another ROI.

CopyTo	-
CreateNew	-
Detach	Detaches the ROI from its parent.
Draw	Draws a EROI in a device context.
DrawFrameWithCurrentPen	-
GetBufferPtr	Retrieves the pointer to the attached pixel buffer.
GetCheckedBufferPtr	Retrieves the pointer to the attached pixel buffer.
IsRoot	Checks if the ROI is a root ROI.
operator=	-
Save	-
Serialize	Serializes the object.

EROI.AsEBaseROI

-

[VB6]

```
EBaseROI AsEBaseROI()  
)  
  
EBaseROI AsEBaseROI()  
)
```

EROI.Attach

Attaches the ROI to another ROI.

[VB6]

```
void Attach(  
    EROI parent  
)
```

Parameters

parent

EROI.Children

Children ROIs.

[VB6]

Children As ()EROI

read-only

EROI.CopyTo

-

[VB6]

```
void CopyTo(  
    EROI other  
)
```

Parameters

other

-

EROI.CreateNew

-

[VB6]

```
EROI CreateNew(  
)
```

EROI.Detach

Detaches the ROI from its parent.

[VB6]

```
void Detach()  
){
```

EROI.Draw

Draws a EROI in a device context.

[VB6]

```
void Draw(  
    EDrawAdapter graphicContext,  
    Single zoomX,  
    Single zoomY,  
    Single panX,  
    Single panY  
)  
  
void Draw(  
    EDrawAdapter graphicContext,  
    EC24Vector c24Vector,  
    Single zoomX,  
    Single zoomY,  
    Single panX,  
    Single panY  
)  
  
void Draw(  
    EDrawAdapter graphicContext,  
    EBW8Vector bw8Vector,  
    Single zoomX,  
    Single zoomY,  
    Single panX,  
    Single panY  
)
```

```

void Draw(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void Draw(
    Long graphicContext,
    EC24Vector c24Vector,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

void Draw(
    Long graphicContext,
    EBW8Vector bw8Vector,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY
)

```

Parameters

graphicContext

Handle to the device context of the destination window.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the image is displayed in **TRUE** scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a **0** value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Pan offset (in pixels) in the horizontal direction. By default, no panning is applied.

panY

Pan offset (in pixels) in the vertical direction. By default, no panning is applied.

c24Vector

When supplied, this parameter allows using a LUT that maps from BW8 to C24 when drawing (false colors).

bw8Vector

When supplied, this parameter allows using a LUT that maps from BW8 to BW8 when drawing.

Remarks

An ROI/image can be drawn (its pixels rendered) using a device context. The horizontal and vertical zooming factors can be different and must be contained in the **1/16..16** range. (MFC users can use the **CDC::GetSafeHdc()** method to obtain a suitable device context handle from a **CDC** instance.)

EROI.DrawFrameWithCurrentPen

-

[VB6]

```
void DrawFrameWithCurrentPen(
    Long graphicContext,
    Boolean bHandles,
    Single zoomX,
    Single zoomY,
    Single x,
    Single y
)
```

Parameters

graphicContext

-

bHandles

-

zoomX

-

zoomY

-

x

-

y

-

EROI.GetBufferPtr

Retrieves the pointer to the attached pixel buffer.

```
[VB6]

Long GetBufferPtr()
)

Long GetBufferPtr(
)

Long GetBufferPtr(
    Long x,
    Long y
)

Long GetBufferPtr(
    Long x,
    Long y
)
```

Parameters

X

Column of the pixel of which we want the address.

Y

Row of the pixel of which we want the address.

Remarks

This function does not check the value of the parameters. Use carefully.

EROI.GetCheckedBufferPtr

Retrieves the pointer to the attached pixel buffer.

```
[VB6]
```

```
Long GetCheckedBufferPtr(
    Long x,
    Long y
)

Long GetCheckedBufferPtr(
    Long x,
    Long y
)
```

Parameters

x

Column of the pixel of which we want the address.

y

Row of the pixel of which we want the address.

EROI.Height

ROI height.

[VB6]

Height As Long

read-write

EROI.IsRoot

Checks if the ROI is a root ROI.

[VB6]

Boolean IsRoot(
 <>)

EROI.IsVoid

Tests whether if the [EROI.GetRootROI@](#) of this hierarchy has a zero size.

[VB6]

IsVoid As Boolean

read-only

EROI.operator=

-

[VB6]

EROI operator=(
 EROI other
 <>)

Parameters

other

EROI.OrgX

Abscissa of the ROI origin in its parent.

[VB6]

OrgX As Long

read-write

EROI.OrgY

Ordinate of the ROI origin in its parent.

[VB6]

OrgY As Long

read-write

EROI.Parent

Parent ROI.

[VB6]

Parent As EROI

read-only

EROI.PixelContainer

Attached pixel container.

[VB6]

PixelContainer As EPixelContainer

read-only

EROI.RowPitch

Buffer row pitch.

[VB6]

RowPitch As Long

read-only

EROI.Save

-

[VB6]

```
void Save(  
    String path,  
    EImageFileType type  
)
```

Parameters

path

-
type

EROI.Serialize

Serializes the object.

```
[VB6]  
void Serialize(  
    ESerializer serializer  
)
```

Parameters

serializer

EROI.TotalOrgX

Abscissa of the ROI origin in the root ROI.

```
[VB6]  
TotalOrgX As Long  
read-only
```

EROI.TotalOrgY

Ordinate of the ROI origin in the root ROI.

```
[VB6]  
TotalOrgY As Long
```

read-only

EROI.Width

ROI width.

[VB6]

Width As Long

read-write

EROIBW1 Class

The EROIBW1 class is used to represent rectangular regions of interest inside BW1 black and white images. See ROIs.

Base Class: [EBaseROI](#)

Derived Class(es): [EImageBW1](#)

PROPERTIES

[FirstSubROI](#)

See GetFirstSubROI in the derived classes

[NextSiblingROI](#)

This method returns the ROI that is the next sibling of this ROI.

Parent	-
TopParent	M
THODS	E
EROIBW1	-
GetBitIndex	-
GetNextROI	See GetNextROI in the derived classes
GetPixel	Allows reading a single pixel value in the ROI or image.
operator=	-
Serialize	-
SetPixel	E Allows writing a single pixel value in the ROI or image. R

OIBW1.EROIBW1

```
[VB6]  
void EROIBW1(  
)  
void EROIBW1(  
EROIBW1 other  
)
```

Parameters

other

EROIBW1.FirstSubROI

See GetFirstSubROI in the derived classes

```
[VB6]  
FirstSubROI As EROIBW1  
read-only
```

EROIBW1.GetBitIndex

```
[VB6]  
Unsupported variant type GetBitIndex(  
Long x,  
Long y  
)
```

Parameters

x

-
Y
-

EROIBW1.GetNextROI

See GetNextROI in the derived classes

```
[VB6]
EROIBW1 GetNextROI(
    EBaseROI startROI
)
```

Parameters

startROI
-

EROIBW1.GetPixel

Allows reading a single pixel value in the ROI or image.

```
[VB6]
EBW1 GetPixel(
    Long x,
    Long y
)
```

Parameters

x
-
Y
-

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the [EBaseROI::GetImagePtr](#) and [EBaseROI::SetImagePtr](#) functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIBW1.NextSiblingROI

This method returns the ROI that is the next sibling of this ROI.

[VB6]

```
NextSiblingROI As EROIBW1  
read-only
```

EROIBW1.operator=

-

```
EROIBW1 operator=(  
    EROIBW1 other  
)
```

Parameters

other

-

EROIBW1.Parent

-

[VB6]

Parent As EROIBW1

read-only

EROIBW1.Serialize

-

[VB6]

```
void Serialize(  
    ESerializer serializer  
)
```

Parameters

serializer

EROIBW1.SetPixel

Allows writing a single pixel value in the ROI or image.

[VB6]

```
void SetPixel(  
    EBW1 value,  
    Long x,  
    Long y  
)
```

Parameters

value

-

x

-

y

-

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the [EBaseROI::GetImagePtr](#) and [EBaseROI::SetImagePtr](#) functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIBW1.TopParent

-

[VB6]

TopParent As EImageBW1

read-only

EROIBW16 Class

The EROIBW16 class is used to represent rectangular regions of interest inside BW16 gray-level images. See ROIs.

Base Class: [EBaseROI](#)

Derived Class(es): [EImageBW16](#)

PROPERTIES

[FirstSubROI](#)

See [GetFirstSubROI](#) in the derived classes

[NextSiblingROI](#)

This method returns the ROI that is the next sibling of this ROI.

[Parent](#)

-

[TopParent](#)

M

E

THODS

[EROIBW16](#)

-

[GetNextROI](#)

See [GetNextROI](#) in the derived classes

GetPixel	Allows reading a single pixel value in the ROI or image.
operator=	-
Serialize	-
SetPixel	Allows writing a single pixel value in the ROI or image. R

OIBW16.EROIBW16

```
[VB6]  
void EROIBW16()  
void EROIBW16(  
    EROIBW16 other  
)
```

Parameters

other

EROIBW16.FirstSubROI

See GetFirstSubROI in the derived classes

[VB6]

FirstSubROI As EROIBW16

read-only

EROIBW16.GetNextROI

See GetNextROI in the derived classes

[VB6]

```
EROIBW16 GetNextROI (
    EBaseROI startROI
)
```

Parameters

startROI

EROIBW16.GetPixel

Allows reading a single pixel value in the ROI or image.

[VB6]

```
EBW16 GetPixel (
    Long x,
    Long y
)
```

Parameters

x

y

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the [EBaseROI::GetImagePtr](#) and [EBaseROI::SetImagePtr](#) functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIBW16.NextSiblingROI

This method returns the ROI that is the next sibling of this ROI.

[VB6]

NextSiblingROI As EROIBW16

read-only

EROIBW16.operator=

[VB6]

```
EROIBW16 operator=(  
    EROIBW16 other  
)
```

Parameters

other

EROIBW16.Parent

-

[VB6]

Parent As EROIBW16

read-only

EROIBW16.Serialize

-

[VB6]

```
void Serialize(  
    ESerializer serializer  
)
```

Parameters

serializer

EROIBW16.SetPixel

Allows writing a single pixel value in the ROI or image.

[VB6]

```
void SetPixel(  
    EBW16 value,  
    Long x,  
    Long y  
)
```

Parameters

value

-

x

-

y

-

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the [EBaseROI::GetImagePtr](#) and [EBaseROI::SetImagePtr](#) functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIBW16.TopParent

-

[VB6]

TopParent As EImageBW16

read-only

EROIBW32 Class

The EROIBW32 class is used to represent rectangular regions of interest inside BW32 gray-level images. See ROIs.

Base Class: [EBaseROI](#)

Derived Class(es): [EImageBW32](#)

PROPERTIES

[FirstSubROI](#)

See [GetFirstSubROI](#) in the derived classes

[NextSiblingROI](#)

This method returns the ROI that is the next sibling of this ROI.

[Parent](#)

-

[TopParent](#)

M

E

THODS

[EROIBW32](#)

-

[GetNextROI](#)

See [GetNextROI](#) in the derived classes

GetPixel	Allows reading a single pixel value in the ROI or image.
operator=	-
Serialize	-
SetPixel	Allows writing a single pixel value in the ROI or image. R

OIBW32.EROIBW32

```
[VB6]  
void EROIBW32()  
void EROIBW32(  
    EROIBW32 other  
)
```

Parameters

other

EROIBW32.FirstSubROI

See GetFirstSubROI in the derived classes

[VB6]

FirstSubROI As EROIBW32

read-only

EROIBW32.GetNextROI

See GetNextROI in the derived classes

[VB6]

```
EROIBW32 GetNextROI (
    EBaseROI startROI
)
```

Parameters

startROI

EROIBW32.GetPixel

Allows reading a single pixel value in the ROI or image.

[VB6]

```
EBW32 GetPixel (
    Long x,
    Long y
)
```

Parameters

x

y

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the [EBaseROI::GetImagePtr](#) and [EBaseROI::SetImagePtr](#) functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIBW32.NextSiblingROI

This method returns the ROI that is the next sibling of this ROI.

[VB6]

NextSiblingROI As EROIBW32

read-only

EROIBW32.operator=

[VB6]

```
EROIBW32 operator=(  
    EROIBW32 other  
)
```

Parameters

other

EROIBW32.Parent

-

[VB6]

Parent As EROIBW32

read-only

EROIBW32.Serialize

-

[VB6]

```
void Serialize(  
    ESerializer serializer  
)
```

Parameters

serializer

EROIBW32.SetPixel

Allows writing a single pixel value in the ROI or image.

[VB6]

```
void SetPixel(  
    EBW32 value,  
    Long x,  
    Long y  
)
```

Parameters

value

-

x

-

y

-

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the [EBaseROI::GetImagePtr](#) and [EBaseROI::SetImagePtr](#) functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIBW32.TopParent

-

[VB6]

TopParent As EImageBW32

read-only

EROIBW8 Class

The EROIBW8 class is used to represent rectangular regions of interest inside BW8 gray-level images. See ROIs.

Base Class: [EBaseROI](#)

Derived Class(es): [EImageBW8](#)

PROPERTIES

[FirstSubROI](#)

See [GetFirstSubROI](#) in the derived classes

[NextSiblingROI](#)

This method returns the ROI that is the next sibling of this ROI.

[Parent](#)

-

[TopParent](#)

M

E

THODS

[EROIBW8](#)

-

[GetNextROI](#)

See [GetNextROI](#) in the derived classes

GetPixel	Allows reading a single pixel value in the ROI or image.
operator=	-
Serialize	-
SetPixel	Allows writing a single pixel value in the ROI or image. R

OIBW8.EROIBW8

```
[VB6]  
void EROIBW8 (  
    )  
void EROIBW8 (  
    EROIBW8 other  
    )
```

Parameters

other

EROIBW8.FirstSubROI

See GetFirstSubROI in the derived classes

[VB6]

FirstSubROI As EROIBW8

read-only

EROIBW8.GetNextROI

See GetNextROI in the derived classes

[VB6]

EROIBW8 GetNextROI(
 EBaseROI startROI
)

Parameters

startROI

EROIBW8.GetPixel

Allows reading a single pixel value in the ROI or image.

[VB6]

EBW8 GetPixel(
 Long x,
 Long y
)

Parameters

x

y

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the [EBaseROI::GetImagePtr](#) and [EBaseROI::SetImagePtr](#) functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIBW8.NextSiblingROI

This method returns the ROI that is the next sibling of this ROI.

[VB6]

NextSiblingROI As EROIBW8

read-only

EROIBW8.operator=

[VB6]

```
EROIBW8 operator=(  
    EROIBW8 other  
)
```

Parameters

other

EROIBW8.Parent

-

[VB6]

Parent As EROIBW8

read-only

EROIBW8.Serialize

-

[VB6]

```
void Serialize(  
    ESerializer serializer  
)
```

Parameters

serializer

EROIBW8.SetPixel

Allows writing a single pixel value in the ROI or image.

[VB6]

```
void SetPixel(  
    EBW8 value,  
    Long x,  
    Long y  
)
```

Parameters

value

-

x

-

y

-

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the [EBaseROI::GetImagePtr](#) and [EBaseROI::SetImagePtr](#) functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIBW8.TopParent

-

[VB6]

TopParent As EImageBW8

read-only

EROIC15 Class

The EROIC15 class is used to represent rectangular regions of interest inside C15 color images. See ROIs.

Base Class: [EBaseROI](#)

Derived Class(es): [EImageC15](#)

PROPERTIES

[FirstSubROI](#)

See GetFirstSubROI in the derived classes

[NextSiblingROI](#)

This method returns the ROI that is the next sibling of this ROI.

[Parent](#)

-

[TopParent](#)

M

E

THODS

[EROIC15](#)

-

[GetNextROI](#)

See GetNextROI in the derived classes

GetPixel	Allows reading a single pixel value in the ROI or image.
operator=	-
Serialize	-
SetPixel	Allows writing a single pixel value in the ROI or image. R

OIC15.EROIC15

```
[VB6]  
void EROIC15(  
)  
void EROIC15(  
    EROIC15 other  
)
```

Parameters

other

EROIC15.FirstSubROI

See GetFirstSubROI in the derived classes

[VB6]

FirstSubROI As EROIC15

read-only

EROIC15.GetNextROI

See GetNextROI in the derived classes

[VB6]

```
EROIC15 GetNextROI(  
    EBaseROI startROI  
)
```

Parameters

startROI

EROIC15.GetPixel

Allows reading a single pixel value in the ROI or image.

[VB6]

```
EC15 GetPixel(  
    Long x,  
    Long y  
)
```

Parameters

x

y

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the [EBaseROI::GetImagePtr](#) and [EBaseROI::SetImagePtr](#) functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIC15.NextSiblingROI

This method returns the ROI that is the next sibling of this ROI.

[VB6]

NextSiblingROI As EROIC15

read-only

EROIC15.operator=

[VB6]

```
EROIC15 operator=(  
    EROIC15 other  
)
```

Parameters

other

EROIC15.Parent

-

[VB6]

Parent As EROIC15

read-only

EROIC15.Serialize

-

[VB6]

```
void Serialize(  
    ESerializer serializer  
)
```

Parameters

serializer

EROIC15.SetPixel

Allows writing a single pixel value in the ROI or image.

[VB6]

```
void SetPixel(  
    EC15 value,  
    Long x,  
    Long y  
)
```

Parameters

value

-

x

-

y

-

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the [EBaseROI::GetImagePtr](#) and [EBaseROI::SetImagePtr](#) functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIC15.TopParent

-

[VB6]

TopParent As EImageC15

read-only

EROIC16 Class

The EROIC16 class is used to represent rectangular regions of interest inside C16 color images. See ROIs.

Base Class: [EBaseROI](#)

Derived Class(es): [EImageC16](#)

PROPERTIES

[FirstSubROI](#)

See GetFirstSubROI in the derived classes

[NextSiblingROI](#)

This method returns the ROI that is the next sibling of this ROI.

[Parent](#)

-

[TopParent](#)

M

E

THODS

[EROIC16](#)

-

[GetNextROI](#)

See GetNextROI in the derived classes

GetPixel	Allows reading a single pixel value in the ROI or image.
operator=	-
Serialize	-
SetPixel	Allows writing a single pixel value in the ROI or image. R

OIC16.EROIC16

```
[VB6]  
void EROIC16(  
)  
void EROIC16(  
    EROIC16 other  
)
```

Parameters

other

EROIC16.FirstSubROI

See GetFirstSubROI in the derived classes

[VB6]

FirstSubROI As EROIC16

read-only

EROIC16.GetNextROI

See GetNextROI in the derived classes

[VB6]

```
EROIC16 GetNextROI(  
    EBaseROI startROI  
)
```

Parameters

startROI

EROIC16.GetPixel

Allows reading a single pixel value in the ROI or image.

[VB6]

```
EC16 GetPixel(  
    Long x,  
    Long y  
)
```

Parameters

x

y

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the [EBaseROI::GetImagePtr](#) and [EBaseROI::SetImagePtr](#) functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIC16.NextSiblingROI

This method returns the ROI that is the next sibling of this ROI.

[VB6]

NextSiblingROI As EROIC16

read-only

EROIC16.operator=

[VB6]

```
EROIC16 operator=(  
    EROIC16 other  
)
```

Parameters

other

EROIC16.Parent

-

[VB6]

Parent As EROIC16

read-only

EROIC16.Serialize

-

[VB6]

```
void Serialize(  
    ESerializer serializer  
)
```

Parameters

serializer

EROIC16.SetPixel

Allows writing a single pixel value in the ROI or image.

[VB6]

```
void SetPixel(  
    EC16 value,  
    Long x,  
    Long y  
)
```

Parameters

value

-

x

-

y

-

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the [EBaseROI::GetImagePtr](#) and [EBaseROI::SetImagePtr](#) functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIC16.TopParent

-

[VB6]

TopParent As EImageC16

read-only

EROIC24 Class

The EROIC24 class is used to represent rectangular regions of interest inside C24 color images. See ROIs.

Base Class: [EBaseROI](#)

Derived Class(es): [EImageC24](#)

PROPERTIES

[FirstSubROI](#)

See [GetFirstSubROI](#) in the derived classes

[NextSiblingROI](#)

This method returns the ROI that is the next sibling of this ROI.

[Parent](#)

-

[TopParent](#)

M

E

THODS

[EROIC24](#)

-

[GetNextROI](#)

See [GetNextROI](#) in the derived classes

GetPixel	Allows reading a single pixel value in the ROI or image.
operator=	-
Serialize	-
SetPixel	E Allows writing a single pixel value in the ROI or image. R

OIC24.EROIC24

```
[VB6]  
void EROIC24(  
)  
void EROIC24(  
    EROIC24 other  
)
```

Parameters

other

EROIC24.FirstSubROI

See GetFirstSubROI in the derived classes

[VB6]

FirstSubROI As EROIC24

read-only

EROIC24.GetNextROI

See GetNextROI in the derived classes

[VB6]

```
EROIC24 GetNextROI(  
    EBaseROI startROI  
)
```

Parameters

startROI

EROIC24.GetPixel

Allows reading a single pixel value in the ROI or image.

[VB6]

```
EC24 GetPixel(  
    Long x,  
    Long y  
)
```

Parameters

x

y

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the [EBaseROI::GetImagePtr](#) and [EBaseROI::SetImagePtr](#) functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIC24.NextSiblingROI

This method returns the ROI that is the next sibling of this ROI.

[VB6]

NextSiblingROI As EROIC24

read-only

EROIC24.operator=

[VB6]

```
EROIC24 operator=(  
    EROIC24 other  
)
```

Parameters

other

EROIC24.Parent

-

[VB6]

Parent As EROIC24

read-only

EROIC24.Serialize

-

[VB6]

```
void Serialize(  
    ESerializer serializer  
)
```

Parameters

serializer

EROIC24.SetPixel

Allows writing a single pixel value in the ROI or image.

[VB6]

```
void SetPixel(  
    EC24 value,  
    Long x,  
    Long y  
)
```

Parameters

value

-

x

-

y

-

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the [EBaseROI::GetImagePtr](#) and [EBaseROI::SetImagePtr](#) functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIC24.TopParent

-

[VB6]

TopParent As EImageC24

read-only

EROIC24A Class

The EROIC24A class is used to represent rectangular regions of interest inside C24A color images. See ROIs.

Base Class: [EBaseROI](#)

Derived Class(es): [EImageC24A](#)

PROPERTIES

[FirstSubROI](#)

See [GetFirstSubROI](#) in the derived classes

[NextSiblingROI](#)

This method returns the ROI that is the next sibling of this ROI.

[Parent](#)

-

[TopParent](#)

M

E

THODS

[EROIC24A](#)

-

[GetNextROI](#)

See [GetNextROI](#) in the derived classes

GetPixel	Allows reading a single pixel value in the ROI or image.
operator=	-
Serialize	-
SetPixel	E Allows writing a single pixel value in the ROI or image. R

OIC24A.EROIC24A

```
[VB6]  
void EROIC24A(  
)  
void EROIC24A(  
    EROIC24A other  
)
```

Parameters

other

EROIC24A.FirstSubROI

See GetFirstSubROI in the derived classes

[VB6]

FirstSubROI As EROIC24A

read-only

EROIC24A.GetNextROI

See GetNextROI in the derived classes

[VB6]

```
EROIC24A GetNextROI (
    EBaseROI startROI
)
```

Parameters

startROI

EROIC24A.GetPixel

Allows reading a single pixel value in the ROI or image.

[VB6]

```
EC24A GetPixel (
    Long x,
    Long y
)
```

Parameters

x

y

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the [EBaseROI::GetImagePtr](#) and [EBaseROI::SetImagePtr](#) functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIC24A.NextSiblingROI

This method returns the ROI that is the next sibling of this ROI.

[VB6]

NextSiblingROI As EROIC24A

read-only

EROIC24A.operator=

[VB6]

```
EROIC24A operator=(  
    EROIC24A other  
)
```

Parameters

other

EROIC24A.Parent

-

[VB6]

Parent As EROIC24A

read-only

EROIC24A.Serialize

-

[VB6]

```
void Serialize(  
    ESerializer serializer  
)
```

Parameters

serializer

EROIC24A.SetPixel

Allows writing a single pixel value in the ROI or image.

[VB6]

```
void SetPixel(  
    EC24A value,  
    Long x,  
    Long y  
)
```

Parameters

value

-

x

-

y

-

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the [EBaseROI::GetImagePtr](#) and [EBaseROI::SetImagePtr](#) functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIC24A.TopParent

-

[VB6]

TopParent As EImageC24A

read-only

EROIC48 Class

The EROIC48 class is used to represent rectangular regions of interest inside C48 color images. See ROIs.

Base Class: [EBaseROI](#)

Derived Class(es): [EImageC48](#)

PROPERTIES

[FirstSubROI](#)

See [GetFirstSubROI](#) in the derived classes

[NextSiblingROI](#)

This method returns the ROI that is the next sibling of this ROI.

[Parent](#)

-

[TopParent](#)

M

E

THODS

[EROIC48](#)

-

[GetNextROI](#)

See [GetNextROI](#) in the derived classes

GetPixel	Allows reading a single pixel value in the ROI or image.
operator=	-
Serialize	-
SetPixel	Allows writing a single pixel value in the ROI or image. R

OIC48.EROIC48

```
[VB6]  
void EROIC48(  
)  
void EROIC48(  
    EROIC48 other  
)
```

Parameters

other

EROIC48.FirstSubROI

See GetFirstSubROI in the derived classes

[VB6]

FirstSubROI As EROIC48

read-only

EROIC48.GetNextROI

See GetNextROI in the derived classes

[VB6]

```
EROIC48 GetNextROI(  
    EBaseROI startROI  
)
```

Parameters

startROI

EROIC48.GetPixel

Allows reading a single pixel value in the ROI or image.

[VB6]

```
EC48 GetPixel(  
    Long x,  
    Long y  
)
```

Parameters

x

y

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the [EBaseROI::GetImagePtr](#) and [EBaseROI::SetImagePtr](#) functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIC48.NextSiblingROI

This method returns the ROI that is the next sibling of this ROI.

[VB6]

NextSiblingROI As EROIC48

read-only

EROIC48.operator=

[VB6]

```
EROIC48 operator=(  
    EROIC48 other  
)
```

Parameters

other

EROIC48.Parent

-

[VB6]

Parent As EROIC48

read-only

EROIC48.Serialize

-

[VB6]

```
void Serialize(  
    ESerializer serializer  
)
```

Parameters

serializer

EROIC48.SetPixel

Allows writing a single pixel value in the ROI or image.

[VB6]

```
void SetPixel(  
    EC48 value,  
    Long x,  
    Long y  
)
```

Parameters

value

-

x

-

y

-

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the [EBaseROI::GetImagePtr](#) and [EBaseROI::SetImagePtr](#) functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIC48.TopParent

-

[VB6]

TopParent As EImageC48

read-only

ERotatedBoundingBox Class

This class represents a rotated bounding box.

Remarks

The rotated bounding box is a rotated, rectangular surface. Its coordinates are floating-point, which makes this class appropriate to handle sub-pixel surfaces.

PROPERTIES

Angle	Returns the angle of the bounding box (in the current angle units).
Center	Returns the coordinate of the center of the bounding box.
CenterX	Returns the abscissa of the center of the bounding box.
CenterY	Returns the ordinate of the center of the bounding box.
Height	Returns the height of the bounding box.
Quadrangle	Returns the coordinates of the four corners of the bounding box.
Width	Returns the width of the bounding box.

METHODS

Draw	Draws the rotated bounding box.
DrawWithCurrentPen	Draws the rotated bounding box.
ERotatedBoundingBox	Constructor of the rotated bounding box.
LocalToGlobalBox	Transforms a (local) rotated bounding box, as another (global) rotated bounding box whose coordinates are defined relatively to the current rotated bounding box.
LocalToGlobalPoint	Transforms a (local) point, as another (global) point whose coordinates are defined relatively to the current rotated bounding box.
operator=	-
Translate	ERotatedBoundingBox.Angle

Returns the angle of the bounding box
(in the current angle units).

[VB6]

Angle As Single

read-only

ERotatedBoundingBox.Center

Returns the coordinate of the center of the bounding box.

[VB6]

Center As EPoint

read-only

ERotatedBoundingBox.CenterX

Returns the abscissa of the center of the bounding box.

[VB6]

CenterX As Single

read-only

ERotatedBoundingBox.CenterY

Returns the ordinate of the center of the bounding box.

[VB6]

CenterY As Single

read-only

ERotatedBoundingBox.Draw

Draws the rotated bounding box.

[VB6]

```
void Draw(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void Draw(
    Long graphicContext,
    ERGBColor color,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)

void Draw(
    EDrawAdapter graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)
```

Parameters

graphicContext

Graphic context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

drawDiagonals

Specifies whether or not the diagonals of the bounding box are drawn.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window.

ERotatedBoundingBox.DrawWithCurrentPen

Draws the rotated bounding box.

[VB6]

```
void DrawWithCurrentPen(
    Long graphicContext,
    Single zoomX,
    Single zoomY,
    Single panX,
    Single panY,
    Boolean drawDiagonals
)
```

Parameters

graphicContext

Graphic context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0** (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

drawDiagonals

Specifies whether or not the diagonals of the bounding box are drawn.

Remarks

Drawing is done in the device context associated to the desired window.

ERotatedBoundingBox.ERotatedBoundingBox

Constructor of the rotated bounding box.

[VB6]

```
void ERotatedBoundingBox(
    Single centerX,
    Single centerY,
    Single width,
    Single height,
    Single angle
)
void ERotatedBoundingBox(
)
void ERotatedBoundingBox(
    ERotatedBoundingBox other
)
```

Parameters

centerX

The abscissa of the center of the bounding box.

centerY

The ordinate of the center of the bounding box.

width

The width of the bounding box.

height

The height of the bounding box.

angle

The angle of the bounding box (in the current angle units).

other

-

ERotatedBoundingBox.Height

Returns the height of the bounding box.

[VB6]

Height As Single

read-only

ERotatedBoundingBox.LocalToGlobalBox

Transforms a (local) rotated bounding box, as another (global) rotated bounding box whose coordinates are defined relatively to the current rotated bounding box.

[VB6]

```
ERotatedBoundingBox LocalToGlobalBox(  
    ERotatedBoundingBox localBox  
)
```

Parameters

localBox

-

ERotatedBoundingBox.LocalToGlobalPoint

Transforms a (local) point, as another (global) point whose coordinates are defined relatively to the current rotated bounding box.

[VB6]

```
EPoint LocalToGlobalPoint(  
    EPoint localPoint  
)
```

Parameters

localPoint

ERotatedBoundingBox.operator=

[VB6]

```
ERotatedBoundingBox operator=(  
    ERotatedBoundingBox other  
)
```

Parameters

other

ERotatedBoundingBox.Quadrangle

Returns the coordinates of the four corners of the bounding box.

[VB6]

```
Quadrangle As EQuadrangle  
read-only
```

ERotatedBoundingBox.Translate

Applies a translation on the center of the bounding box.

[VB6]

```
void Translate(  
    Single offsetX,  
    Single offsetY  
)
```

Parameters

offsetX

The offset along the X-axis.

offsetY

The offset along the Y-axis.

ERotatedBoundingBox.Width

Returns the width of the bounding box.

[VB6]

Width As Single

read-only

ESearchParamsType Class

This class is instantiated once in each [EMatrixCodeReader](#) and represents the search parameters that are explored when reading a [EMatrixCode](#).

Remarks

This class contains 4 sets of search parameters that are scanned at read time. At [EMatrixCodeReader](#) construction time, these sets of values are initialized with all possible values. This means, for instance, that a data matrix code read in a freshly created reader is matched against all possible logical sizes. As a consequence, the default values of these sets are not repeated here. They are assumed to represent every possible search parameters. The [ESearchParamsType](#) object needs to be used only if you wish to "override" the learning process.

PROPERTIES[ContrastCount](#)

The size of the list of [EMatrixCodeContrastMode](#) the candidate is matched against at read time.

[FamilyCount](#)

The size of the list of [EFamily](#) the candidate is matched against at read time.

[FlippingCount](#)

The size of the list of [EFlipping](#) the candidate is matched against at read time.

[LogicalSizeCount](#)

M The size of the list of [ELogicalSize](#) the candidate is matched against at read time.

E

THODS[AddContrast](#)

Adds a new contrast mode to the list of [EMatrixCodeContrastMode](#) the candidate is matched against at read time.

[AddFamily](#)

Adds a new family to the list of [EFamily](#) the candidate is matched against at read time.

[AddFlipping](#)

Adds a new flipping to the list of [EFlipping](#) the candidate is matched against at read time.

[AddLogicalSize](#)

Adds a new logical size to the list of [ELogicalSize](#) the candidate is matched against at read time.

[ClearContrast](#)

Clears the list of contrast modes the candidate is matched against at read time.

[ClearFamily](#)

Clears the list of families the candidate is matched against at read time.

[ClearFlipping](#)

Clears the list of flippings the candidate is matched against at read time.

[ClearLogicalSize](#)

Clears the list of logical sizes the candidate is matched against at read time.

[GetContrast](#)

Gets an item in the list of [EMatrixCodeContrastMode](#) the candidate is matched against at read time.

[GetFamily](#)

Gets an item in the list of [EFamily](#) the candidate is matched against at read time.

[GetFlipping](#)

Gets an item in the list of [EFlipping](#) the candidate is matched against at read time.

[GetLogicalSize](#)

Gets an item in the list of [ELogicalSize](#) the candidate is matched against at read time.

[RemoveContrast](#)

Removes the contrast mode from the list of [EMatrixCodeContrastMode](#) the candidate is matched against at read time.

[RemoveFamily](#)

Removes the family from the list of [EFamily](#) the candidate is matched against at read time.

[RemoveFlipping](#)

Removes the flipping from the list of [EFlipping](#) the candidate is matched against at read time.

[RemoveLogicalSize](#)

Removes the logical size from the list of [ELogicalSize](#) the candidate is matched against at read time.
S

earchParamsType.Ad
dContrast

Adds a new contrast mode to the list of [EMatrixCodeContrastMode](#) the candidate is matched against at read time.

[VB6]

```
void AddContrast(
    EMATRIXCODECONTRASTMODE searchContrast
)
```

Parameters

searchContrast

Contrast mode to add to the list.

ESearchParamsType.AddFamily

Adds a new family to the list of [EFamily](#) the candidate is matched against at read time.

[VB6]

```
void AddFamily(
    EFamily searchFamily
)
```

Parameters

searchFamily

Family to add to the list.

ESearchParamsType.AddFlipping

Adds a new flipping to the list of [EFlipping](#) the candidate is matched against at read time.

[VB6]

```
void AddFlipping(
    EFlipping searchFlipping
)
```

Parameters

searchFlipping

Flipping to add to the list.

ESearchParamsType.AddLogicalSize

Adds a new logical size to the list of [ELogicalSize](#) the candidate is matched against at read time.

[VB6]

```
void AddLogicalSize(  
    ELogicalSize searchLogicalSize  
)
```

Parameters

searchLogicalSize

Logical size to add to the list.

ESearchParamsType.ClearContrast

Clears the list of contrast modes the candidate is matched against at read time.

[VB6]

```
void ClearContrast(  
)
```

ESearchParamsType.ClearFamily

Clears the list of families the candidate is matched against at read time.

[VB6]

```
void ClearFamily()  
)
```

ESearchParamsType.ClearFlipping

Clears the list of flippings the candidate is matched against at read time.

[VB6]

```
void ClearFlipping()  
)
```

ESearchParamsType.ClearLogicalSize

Clears the list of logical sizes the candidate is matched against at read time.

[VB6]

```
void ClearLogicalSize()  
)
```

ESearchParamsType.CtrastCount

The size of the list of [EMatrixCodeContrastMode](#) the candidate is matched against at read time.

[VB6]

ContrastCount As Long

read-only

ESearchParamsType.FamilyCount

The size of the list of [EFamily](#) the candidate is matched against at read time.

[VB6]

FamilyCount As Long

read-only

ESearchParamsType.FlippingCount

The size of the list of [EFlipping](#) the candidate is matched against at read time.

[VB6]

FlippingCount As Long

read-only

ESearchParamsType.GetContrast

Gets an item in the list of [EMatrixCodeContrastMode](#) the candidate is matched against at read time.

[VB6]

```
EMatrixCodeContrastMode GetContrast(
    Long index
)
```

Parameters

index

Position in the list.

ESearchParamsType.GetFamily

Gets an item in the list of [EFamily](#) the candidate is matched against at read time.

[VB6]

```
EFamily GetFamily(
    Long index
)
```

Parameters

index

Position in the list.

ESearchParamsType.GetFlipping

Gets an item in the list of [EFlipping](#) the candidate is matched against at read time.

[VB6]

```
EFlipping GetFlipping(
    Long index
)
```

Parameters

index

Position in the list.

ESearchParamsType.GetLogicalSize

Gets an item in the list of [ELogicalSize](#) the candidate is matched against at read time.

[VB6]

```
ELogicalSize GetLogicalSize(  
    Long index  
)
```

Parameters

index

Position in the list.

ESearchParamsType.LogicalSizeCount

The size of the list of [ELogicalSize](#) the candidate is matched against at read time.

[VB6]

```
LogicalSizeCount As Long  
read-only
```

ESearchParamsType.RemoveContrast

Removes the contrast mode from the list of [EMatrixCodeContrastMode](#) the candidate is matched against at read time.

[VB6]

```
void RemoveContrast(
    EMATRIXCODECONTRASTMODE searchContrast
)
```

Parameters

searchContrast

Contrast mode to remove from the list.

ESearchParamsType.RemoveFamily

Removes the family from the list of [EFamily](#) the candidate is matched against at read time.

[VB6]

```
void RemoveFamily(
    EFamily searchFamily
)
```

Parameters

searchFamily

Family to remove from the list.

ESearchParamsType.RemoveFlipping

Removes the flipping from the list of [EFlipping](#) the candidate is matched against at read time.

[VB6]

```
void RemoveFlipping(
    EFlipping searchFlipping
)
```

Parameters

searchFlipping

Flipping to remove from the list.

ESearchParamsType.RemoveLogicalSize

Removes the logical size from the list of [ELogicalSize](#) the candidate is matched against at read time.

[VB6]

```
void RemoveLogicalSize(
    ELogicalSize searchLogicalSize
)
```

Parameters

searchLogicalSize

Logical size to remove from the list.

ESerializer Class

Abstract interface for file-like objects.

Remarks

The ESerializer object manages operations of reading from and writing to an archive (a file on the system hard disk, for instance). ESerializer objects cannot be instantiated directly. To create an ESerializer object, one of the following static factory methods has to be used:

Note. An ESerializer object can not be used in the same time for reading and writing. So, [ESerializer::CreateFileWriter](#) creates an ESerializer object that should be used with **Save** methods and [ESerializer::CreateFileReader](#) creates an ESerializer object that should be used with **Load** methods.

Derived Class(es): [EFilePointerSerializer](#) [EFileSerializer](#)

PROPERTIES**Writing**

M Returns **TRUE** if the ESerializer object has been created for writing and **FALSE** otherwise.

E

THODS**Close**

Closes the file associated with the ESerializer object.

CreateCallbackReader

-

CreateCallbackWriter

-

CreateFileReader

Returns an ESerializer object suitable for reading from a file.

CreateFileWriter**ESerializer.Close**

Retires the ESerializer object and releases all its resources.

Closes the file associated with the ESerializer object.

```
[VB6]  
void Close()  
)
```

ESerializer.CreateCallbackReader

-

```
[VB6]  
ESerializer CreateCallbackReader(  
    Long isEOS,  
    Long setCurrentPos,  
    Long getCurrentPos,  
    Long serializeMemory,  
    Long close,  
    Long cookie  
)
```

Parameters

isEOS

-

setCurrentPos

-

getCurrentPos

-

serializeMemory

-

close

-

cookie

-

ESerializer.CreateCallbackWriter

-

[VB6]

```
ESerializer CreateCallbackWriter(  
    Long isEOS,  
    Long setCurrentPos,  
    Long getCurrentPos,  
    Long serializeMemory,  
    Long close,  
    Long cookie  
)
```

Parameters

isEOS

-

setCurrentPos

-

getCurrentPos

-

serializeMemory

-

close

-

cookie

-

ESerializer.CreateFileReader

Returns an ESerializer object suitable for reading from a file.

[VB6]

```
ESerializer CreateFileReader(
    String filePath
)
```

Parameters

filePath

Full path and name specification of the file to be used to create the ESerializer object.

Remarks

It is up to users to delete the ESerializer object when they have done using it in **Load** calls. If the call does not succeed, it returns **NULL**. Please check the Open eVision error code to get further informations.

ESerializer.CreateFileWriter

Returns an ESerializer object suitable for opening a file and writing into it.

[VB6]

```
ESerializer CreateFileWriter(
    String filePath,
    ESerializer.FileWriterMode mode
)
```

Parameters

filePath

Full path and name specification of the file to be used to create the ESerializer object.

mode

Creation mode of the storage file, as defined by [ESerializer.FileWriterMode](#) (by default, **Create**).

Remarks

The `ESerializerFileWriterMode` parameter is an enumerated type that allows to control what happens when the file already exists: * If `mode` is **Create**, the call will not succeed if the file already exists. * If `mode` is **Overwrite**, the existing file will be overwritten. * If `mode` is **Append**, the new data will be appended to the existing file content. It is up to users to delete the `ESerializer` object when they have done using it in **Save** calls. If the call does not succeed, it returns **NULL**. Please check the Open eVision error code to get further information.

ESerializer.Writing

Returns **TRUE** if the `ESerializer` object has been created for writing and **FALSE** otherwise.

[VB6]

Writing As Boolean

read-only

EShape Class

Abstract class to federate the classes that can be hierarchically attached together (from a geometrical point of view).

Derived Class(es): `ERectangleShape` `ECircleShape` `EFrameShape` `ELineShape` `EPointShape` `EWedgeShape` `EWorldShape`

PROPERTIES

Active

Flag indicating whether the shape is active or not.

ActiveRecursive

Sets the flag indicating whether the shape and all its daughters are active or not.

ActualShape	Flag indicating whether an inquiry returns a result pertaining to the nominal gauge (FALSE , default) or the fitted model (TRUE).
ActualShapeRecursive	Sets the flag indicating whether an inquiry returns a result pertaining to the nominal gauge (FALSE , default) or the fitted model (TRUE). The flag is set in this shape and all its daughters.
ClosestShape	Closest shape among the daughters.
Dragable	Flag indicating whether the shape can be dragged or not.
DragableRecursive	Sets the flag indicating whether the shape and all its daughters can be dragged or not.
HitHandle	Handle currently under the cursor.
HitShape	Pointer to the shape currently under the cursor.
Labeled	Flag indicating whether the shape label should be displayed or not.
LabeledRecursive	Sets the flag indicating whether the shape label should be displayed or not. The flag is set in this shape and all its daughters.
Mother	Pointer to the mother shape.

Name	Name of the EShape .
NumDaughters	Number of daughters attached to the shape.
PanX	Current horizontal panning factor for drawing operations.
PanY	Current vertical panning factor for drawing operations.
Resizable	Flag indicating whether the shape can be resized or not.
ResizableRecursive	Sets the flag indicating whether the shape and all its daughters can be resized or not.
Rotatable	Flag indicating whether the shape can be rotated or not.
RotatableRecursive	Sets the flag indicating whether the shape and all its daughters can be rotated or not.
Selectable	Flag indicating whether the shape can be selected or not.
SelectableRecursive	Sets the flag indicating whether the shape and all its daughters can be selected or not.

Selected	Flag indicating whether the shape is selected or not.
SelectedRecursive	Sets the flag indicating whether the shape and all its daughters are selected or not.
Type	Shape type.
Visible	Flag indicating whether the shape is visible or not.
VisibleRecursive	Sets the flag indicating whether the shape and its daughter shapes are visible or not.
WorldShape	-
ZoomX	Current horizontal zooming factor for drawing operations.
ZoomY	<p>M Current vertical zooming factor for drawing operations.</p> <p>E</p>
THODS	
Attach	Attaches the gauge to a mother gauge or shape.
Closest	Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use EShape::ClosestShape .

Detach	Detaches the gauge from its mother gauge or shape.
DetachDaughters	Detaches the daughter gauges or shapes.
DisableBehaviorFilter	Disables (i.e. removes) a condition from the list of conditions in the behavior filter.
DisableTypeFilter	-
Drag	-
Draw	Draws a graphical representation of a shape, as defined by EDrawingMode .
DrawWithCurrentPen	Draws a graphical representation of a shape, as defined by EDrawingMode .
EnableBehaviorFilter	Modifies the so-called behavior filter that is a conjunction of conditions specifying when a shape or a gauge should be displayed or taken into consideration when monitoring the mouse interactions.
EnableTypeFilter	-
GetAllocated	-

GetDaughter	Returns a pointer to the specified daughter gauge or shape.
GetDraggingMode	-
GetOptionalDraw	-
GetShapeNamed	Returns a pointer to a daughter gauge or shape specified by its name.
HitTest	-
InvalidateWorld	-
Load	-
LocalToSensor	-
Process	-
Save	-
SensorToLocal	-
SetAllocated	-

[SetCursor](#)

Sets the cursor current coordinates.

[SetDraggingMode](#)

-

[SetOptionalDraw](#)

-

[SetPan](#)

Sets the horizontal and vertical panning factors for drawing operations.

[SetZoom](#)

EShape.Active

Gets or sets the flag indicating whether the shape is active or not.

Flag indicating whether the shape is active or not.

[VB6]

Active As Boolean

read-write

EShape.ActiveRecursive

Sets the flag indicating whether the shape and all its daughters are active or not.

[VB6]

ActiveRecursive As Boolean

read-write

Remarks

When complex gauging is required, several gauges can be grouped together. Applying [EShape::Process](#) to the mother gauge or shape triggers the measurement of the whole. Only the active gauges will participate in the process.

EShape.ActualShape

Flag indicating whether an inquiry returns a result pertaining to the nominal gauge (**FALSE**, default) or the fitted model (**TRUE**).

[VB6]

ActualShape As Boolean

read-write

EShape.ActualShapeRecursive

Sets the flag indicating whether an inquiry returns a result pertaining to the nominal gauge (**FALSE**, default) or the fitted model (**TRUE**). The flag is set in this shape and all its daughters.

[VB6]

ActualShapeRecursive As Boolean

read-write

EShape.Attach

Attaches the gauge to a mother gauge or shape.

[VB6]

```
void Attach(  
    EShape *mother  
)
```

Parameters

mother

Pointer to the mother gauge or shape.

Remarks

When attached to a mother gauge, be aware that daughter gauges are not positioned according to the nominal mother gauge position, but to its corresponding fitted model.

EShape.Closest

Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use [EShape::ClosestShape](#).

[VB6]

```
void Closest(  
)
```

EShape.ClosestShape

Closest shape among the daughters.

[VB6]

ClosestShape As EShape

read-only

Remarks

Use [EShape::Closest](#) to recompute the closest shape.

EShape.Detach

Detaches the gauge from its mother gauge or shape.

[VB6]

```
void Detach()  
{}
```

EShape.DetachDaughters

Detaches the daughter gauges or shapes.

[VB6]

```
void DetachDaughters()  
{}
```

EShape.DisableBehaviorFilter

Disables (i.e. removes) a condition from the list of conditions in the behavior filter.

```
[VB6]  
  
void DisableBehaviorFilter(  
    EShapeBehavior behavior  
)
```

Parameters

behavior

The behavior of the shape to be removed from the behavior filter.

Remarks

The condition to be disabled is identified by the behavior about which the condition is. Disabling a behavior leads to less restrictive conditions for the **Draw** and **HitTest** methods to be actually carried on.

EShape.DisableTypeFilter

```
[VB6]  
void DisableTypeFilter()  
){
```

EShape.Drag

```
[VB6]  
void Drag(  
    Long n32CursorX,  
    Long n32CursorY  
)
```

Parameters

n32CursorX

-

n32CursorY

-

EShape.Dragable

Flag indicating whether the shape can be dragged or not.

[VB6]

Dragable As Boolean

read-write

EShape.DragableRecursive

Sets the flag indicating whether the shape and all its daughters can be dragged or not.

[VB6]

DragableRecursive As Boolean

read-write

EShape.Draw

Draws a graphical representation of a shape, as defined by [EDrawingMode](#).

[VB6]

```
void Draw(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)

void Draw(
    Long graphicContext,
    ERGBColor color,
    EDrawingMode drawingMode,
    Boolean daughters
)

void Draw(
    EDrawAdapter graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by [EDrawingMode](#).

daughters

TRUE if the daughters gauges are to be displayed also.

color

EShape.DrawWithCurrentPen

Draws a graphical representation of a shape, as defined by [EDrawingMode](#).

[VB6]

```
void DrawWithCurrentPen(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

Parameters*graphicContext*

Handle of the device context on which to draw.

*drawingMode*Indicates how the point location or model fitting gauge must be displayed, as defined by [EDrawingMode](#).*daughters***TRUE** if the daughters gauges are to be displayed also.

EShape.EnableBehaviorFilter

Modifies the so-called **behavior filter** that is a conjunction of conditions specifying when a shape or a gauge should be displayed or taken into consideration when monitoring the mouse interactions.

[VB6]

```
void EnableBehaviorFilter(
    EShapeBehavior behavior,
    Boolean value
)
```

Parameters*behavior*

The behavior of the shape to be tested.

*value*The value at which the behavior property should be set to pass the test. By default, equals **True**.

Remarks

This method registers a new necessary condition for the **Draw** and **HitTest** families of methods to be actually carried on. Such a condition is about the behavior of the shape, as specified by the **behavior** argument. Initially, the behavior filter contains an empty list of conditions, which means that the **Draw** and **HitTest** methods will always be executed. Adding a new condition through **EShape::EnableBehaviorFilter** will introduce a new restriction on the effective execution of these methods. Use **EShape::DisableBehaviorFilter** to remove a condition from the behavior filter.

EShape.EnableTypeFilter

-

[VB6]

```
void EnableTypeFilter(
    Long un32Types
)
```

Parameters

un32Types

-

EShape.GetAllocated

-

[VB6]

```
Boolean GetAllocated(
)
```

EShape.GetDaughter

Returns a pointer to the specified daughter gauge or shape.

[VB6]

```
EShape GetDaughter(  
    Long index  
)
```

Parameters

index

Daughter gauge or shape index.

EShape.GetDraggingMode

-

[VB6]

```
EDraggingMode GetDraggingMode(  
)
```

EShape.GetOptionalDraw

-

[VB6]

```
Boolean GetOptionalDraw(  
)
```

EShape.GetShapeNamed

Returns a pointer to a daughter gauge or shape specified by its name.

[VB6]

```
EShape GetShapeNamed(  
    String name  
)
```

Parameters

name

Name of the daughter gauge or shape.

EShape.HitHandle

Handle currently under the cursor.

[VB6]

```
HitHandle As EDragHandle  
read-only
```

Remarks

When the cursor is over a particular handle, its shape could be changed for feedback.

EShape.HitShape

Pointer to the shape currently under the cursor.

[VB6]

HitShape As EShape

read-only

Remarks

When the cursor is over a particular shape, its shape could be changed for feedback.

EShape.HitTest

-

[VB6]

```
Boolean HitTest(  
    Boolean bDaughters  
)
```

Parameters

bDaughters

-

EShape.InvalidateWorld

-

[VB6]

```
void InvalidateWorld(  
)
```

EShape.Labeled

Flag indicating whether the shape label should be displayed or not.

[VB6]

Labeled As Boolean

read-write

EShape.LabeledRecursive

Sets the flag indicating whether the shape label should be displayed or not. The flag is set in this shape and all its daughters.

[VB6]

LabeledRecursive As Boolean

read-write

EShape.Load

-

[VB6]

```
void Load(  
    ESerializer serializer,  
    Boolean daughters  
)
```

```
void Load(
    String stream,
    Boolean daughters
)
```

Parameters

serializer

-

daughters

-

stream

-

EShape.LocalToSensor

-

[VB6]

```
EPoint LocalToSensor(
    EPoint LPoint
)
```

Parameters

LPoint

-

EShape.Mother

Pointer to the mother shape.

[VB6]

Mother As EShape

read-only

EShape.Name

Name of the [EShape](#).

[VB6]

Name As String

read-write

EShape.NumDaughters

Number of daughters attached to the shape.

[VB6]

NumDaughters As Long

read-only

EShape.PanX

Current horizontal panning factor for drawing operations.

[VB6]

PanX As Single

read-only

EShape.PanY

Current vertical panning factor for drawing operations.

[VB6]

PanY As Single

read-only

EShape.Process

-

[VB6]

```
void Process(
    EROIBW8 pSrc,
    Boolean bDaughters
)
```

Parameters

pSrc

-

bDaughters

-

EShape.Resizable

Flag indicating whether the shape can be resized or not.

[VB6]

Resizable As Boolean

read-write

EShape.ResizableRecursive

Sets the flag indicating whether the shape and all its daughters can be resized or not.

[VB6]

ResizableRecursive As Boolean

read-write

EShape.Rotatable

Flag indicating whether the shape can be rotated or not.

[VB6]

Rotatable As Boolean

read-write

EShape.RotatableRecursive

Sets the flag indicating whether the shape and all its daughters can be rotated or not.

[VB6]

RotatableRecursive As Boolean

read-write

EShape.Save

-

[VB6]

```
void Save(
    ESerializer serializer,
    Boolean daughters
)

void Save(
    String stream,
    Boolean daughters
)
```

Parameters

serializer

-

daughters

-

stream

-

EShape.Selectable

Flag indicating whether the shape can be selected or not.

[VB6]

Selectable As Boolean

read-write

EShape.SelectableRecursive

Sets the flag indicating whether the shape and all its daughters can be selected or not.

[VB6]

SelectableRecursive As Boolean

read-write

EShape.Selected

Flag indicating whether the shape is selected or not.

[VB6]

Selected As Boolean

read-write

EShape.SelectedRecursive

Sets the flag indicating whether the shape and all its daughters are selected or not.

[VB6]

SelectedRecursive As Boolean

read-write

EShape.SensorToLocal

-

[VB6]

```
EPoint SensorToLocal(
    EPoint SPoint
)
```

Parameters

SPoint

-

EShape.SetAllocated

-

[VB6]

```
void SetAllocated(  
    Boolean bAllocated,  
    Boolean bDaughters  
)
```

Parameters

bAllocated

-

bDaughters

-

EShape.SetCursor

Sets the cursor current coordinates.

[VB6]

```
void SetCursor(  
    Long x,  
    Long y  
)
```

Parameters

x

Cursor current coordinates.

y

Cursor current coordinates.

EShape.SetDraggingMode

[VB6]

```
void SetDraggingMode(  
    EDraggingMode eDraggingMode,  
    Boolean bDaughters  
)
```

Parameters

eDraggingMode

-

bDaughters

-

EShape.SetOptionalDraw

[VB6]

```
void SetOptionalDraw(  
    Boolean bFound,  
    Boolean bDaughters  
)
```

Parameters

bFound

-

bDaughters

-

EShape.SetPan

Sets the horizontal and vertical panning factors for drawing operations.

[VB6]

```
void SetPan(
    Single f32PanX,
    Single f32PanY
)
```

Parameters

f32PanX

-

f32PanY

-

Remarks

All objects attached to an [EWorldShape](#) object inherit of the same panning factor.

EShape.SetZoom

Sets the horizontal and vertical zooming factors for drawing operations.

[VB6]

```
void SetZoom(
    Single f32ZoomX,
    Single f32ZoomY
)
```

Parameters

f32ZoomX

-

f32ZoomY

-

Remarks

All objects attached to an [EWorldShape](#) inherit of the same zooming factor.

EShape.Type

Shape type.

[VB6]

Type As EShapeType

read-only

EShape.Visible

Flag indicating whether the shape is visible or not.

[VB6]

Visible As Boolean

read-write

EShape.VisibleRecursive

Sets the flag indicating whether the shape and its daughter shapes are visible or not.

[VB6]

VisibleRecursive As Boolean

read-write

EShape.WorldShape

-

[VB6]

WorldShape As EWorldShape

read-only

EShape.ZoomX

Current horizontal zooming factor for drawing operations.

[VB6]

ZoomX As Single

read-only

EShape.ZoomY

Current vertical zooming factor for drawing operations.

[VB6]

ZoomY As Single

read-only

EThreeLayersImageSegmenter Class

The base class from which all the segmenters that produce three layers derive.

Remarks

The Layer Encoding can be enabled/disabled for each layer individually. The index of the layers in the coded image can also be assigned individually.

Base Class: [EImageSegmenter](#)

Derived Class(es): [EGrayscaleDoubleThresholdSegmenter](#)

PROPERTIES

BlackLayerEncoded	Black layer encoding status.
BlackLayerIndex	Index of the black layer in the destination coded image.
NeutralLayerEncoded	Neutral layer encoding status.
NeutralLayerIndex	Index of the neutral layer in the destination coded image.
WhiteLayerEncoded	White layer encoding status.
WhiteLayerIndex	Index of the white layer in the destination coded image.

EThreeLayersImageSegmenter.BlackLayerEncoded

Black layer encoding status.

[VB6]

BlackLayerEncoded As Boolean

read-write

EThreeLayersImageSegmenter.BlackLayerIndex

Index of the black layer in the destination coded image.

[VB6]

BlackLayerIndex As Long

read-write

Remarks

Setting this property automatically switches on the encoding of the black layer.

EThreeLayersImageSegmenter.NeutralLayerEncoded

Neutral layer encoding status.

[VB6]

NeutralLayerEncoded As Boolean

read-write

EThreeLayersImageSegmenter.NeutralLayerIndex

Index of the neutral layer in the destination coded image.

[VB6]

NeutralLayerIndex As Long

read-write

Remarks

Setting this property automatically switches on the encoding of the neutral layer.

EThreeLayersImageSegmenter.WhiteLayerEncoded

White layer encoding status.

[VB6]

WhiteLayerEncoded As Boolean

read-write

EThreeLayersImageSegmenter.WhiteLayerIndex

Index of the white layer in the destination coded image.

[VB6]

WhiteLayerIndex As Long

read-write

Remarks

Setting this property automatically switches on the encoding of the white layer.

ETwoLayersImageSegmenter Class

The base class from which all the segmenters that produce two layers derive.

Remarks

The Layer Encoding can be enabled/disabled for each layer individually. The index of the layers in the coded image can also be assigned individually.

Base Class: [EImageSegmenter](#)

Derived Class(es): [EBinaryImageSegmenter](#) [EColorRangeThresholdSegmenter](#)
[EColorSingleThresholdSegmenter](#) [EGrayscaleSingleThresholdSegmenter](#)
[EImageRangeSegmenter](#) [EReferenceImageSegmenter](#)

PROPERTIES

BlackLayerEncoded	Black layer encoding status.
BlackLayerIndex	Index of the black layer in the destination coded image.
WhiteLayerEncoded	White layer encoding status.
WhiteLayerIndex	Index of the white layer in the destination coded image.

ETwoLayersImageSegmenter.BlackLayerEncoded

Black layer encoding status.

[VB6]

BlackLayerEncoded As Boolean

read-write

ETwoLayersImageSegmenter.BlackLayerIndex

Index of the black layer in the destination coded image.

[VB6]

BlackLayerIndex As Long

read-write

Remarks

Setting this property automatically switches on the encoding of the black layer.

ETwoLayersImageSegmenter.WhiteLayerEncoded

White layer encoding status.

[VB6]

WhiteLayerEncoded As Boolean

read-write

ETwoLayersImageSegmenter.WhiteLayerIndex

Index of the white layer in the destination coded image.

[VB6]

WhiteLayerIndex As Long

read-write

Remarks

Setting this property automatically switches on the encoding of the white layer.

EUnwarpingLut Class

This class is used only as a lookup table in the [EWorldShape::Unwarp](#) and [EWorldShape::SetupUnwarp](#) methods. It has no other use of its own.

METHODS

EUnwarpingLut

Constructs a EUnwarpingLut object that is used to speed-up the unwarping process.

U

nwarpingLut.EUnwarpingLut

Constructs a EUnwarpingLut object that is used to speed-up the unwarping process.

[VB6]

```

void EUnwarpingLut(
    EUnwarpingLut other
)
void EUnwarpingLut(
)

```

Parameters*other*

-

EVector Class

Base class for all typed vectors.

Remarks

This class contains all methods that are not type specific. Mainly methods to handle elements count and serialization

Derived Class(es): [EBW32Vector](#) [EBW16PathVector](#) [EBW16Vector](#) [EBW8PathVector](#) [EBW8Vector](#) [EWHistogramVector](#) [EC24PathVector](#) [EC24Vector](#) [EPathVector](#) [EColorVector](#) [EPeakVector](#)

PROPERTIES

NumElements**M**

Number of elements in the vector.

E

THODS

Empty

Resets the number of elements to **0**.

RemoveElement

Removes the element at the specified position

Serialize



EVector.Empty

Resets the number of elements to **0**.

[VB6]

```
void Empty()  
)
```

EVector.NumElements

Number of elements in the vector.

[VB6]

NumElements As Long
read-write

EVector.RemoveElement

Removes the element at the specified position

[VB6]

```
void RemoveElement(
    Long index
)
```

Parameters

index

Index, between **0** and [EVector::NumElements](#) (excluded) of the element to be accessed.

EVector.Serialize

```
[VB6]
void Serialize(
    ESerializer serializer,
    Long un32Version
)
```

Parameters

serializer

-
un32Version

EWedge Class

Represents a model of a wedge (disk, ring, sector or curvilinear quadrilateral) in EasyGauge.

Base Class: [EFrame](#)

PROPERTIES

Amplitude	Angular amplitude of the EWedge .
ApexAngle	Angular position at the apex of the EWedge .
Breadth	Breadth of the EWedge .
Direct	-
EndAngle	Ending angular position of the EWedge .
FullBreadth	Flag indicating if the EWedge has a full breadth (breadth = radius) .
FullCircle	Flag indicating if the EWedge is a full circle (amplitude = 2 PI) .
InnerApex	Inner apex point coordinates of the EWedge .
InnerArcLength	Inner circle arc length of the EWedge .
InnerDiameter	Inner diameter of the EWedge .
InnerEnd	Inner end point coordinates of the EWedge .

InnerOrg	Inner origin point coordinates of the EWedge .
InnerRadius	Inner radius of the EWedge
OrgAngle	Angular position from where the EWedge extents.
OuterApex	Outer apex point coordinates of the EWedge .
OuterArcLength	Outer circle arc length of the EWedge .
OuterDiameter	Outer diameter of the EWedge .
OuterEnd	Outer end point coordinates of the EWedge .
OuterOrg	Outer origin point coordinates of the EWedge .
OuterRadius	M Outer radius of the EWedge .
THODS	
CopyTo	Copies all the data of the current EWedge object into another EWedge object and returns it.
EWedge	Constructs a EWedge object.

GetCorners	Retrieves the coordinates of each corner of the EWedge .
GetEdges	Retrieves each edge of the EWedge .
GetInnerPoint	Returns the coordinates of a particular point specified by its location along the inner circle arc.
GetMidEdges	Retrieves the center coordinates of each edge of the wedge fitting gauge.
GetOuterPoint	Returns the coordinates of a particular point specified by its location along the outer circle arc.
GetPoint	Returns the coordinates of a particular point specified by its location along the wedge perimeter.
operator=	Copies all the data from another EWedge object into the current EWedge object
SetDiameters	Sets the nominal inner/outer diameter and breadth of the EWedge .
SetFromCenterAndOrigin	Sets the geometric parameters (nominal position and diameter, breadth, angular position and amplitude) of an EWedge object.

[SetFromOriginMiddleEnd](#)

Sets the geometric parameters (nominal position and diameter, breadth, angular position and amplitude) of an EWedge object.

[SetFromTwoPoints](#)

-

[SetRadii](#)

E Sets the nominal radius and breadth of the EWedge.

W

edge.Amplitude

Angular amplitude of the [EWedge](#).

[VB6]

Amplitude As Single

read-write

Remarks

A EWedge is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance. The sign of the rotation angle depends whether the field of view is calibrated or not. * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value. * When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

EWedge.ApexAngle

Angular position at the apex of the [EWedge](#).

[VB6]

ApexAngle As Single

read-only

Remarks

A EWedge is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance. The sign of the rotation angle depends whether the field of view is calibrated or not. * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value. * When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

EWedge.Breadth

Breadth of the [EWedge](#).

[VB6]

Breadth As Single

read-only

Remarks

A EWedge is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.

EWedge.CopyTo

Copies all the data of the current EWedge object into another EWedge object and returns it.

[VB6]

```
EWedge CopyTo(  
    EWedge destinationImage  
)
```

Parameters

destinationImage

Pointer to the EWedge object in which the current EWedge object data have to be copied.

Remarks

In case of a **NULL** pointer, a new EWedge object will be created and returned.

EWedge.Direct

-

[VB6]

Direct As Boolean

read-write

EWedge.EndAngle

Ending angular position of the **EWedge**.

[VB6]

EndAngle As Single

read-only

Remarks

A EWedge is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance. The sign of the rotation angle depends whether the field of view is calibrated or not. * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value. * When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

EWedge.EWedge

Constructs a EWedge object.

[VB6]

```
void EWedge()
)

void EWedge(
    EPoint center,
    Single diameter,
    Single breadth,
    Single originAngle,
    Boolean direct
)

void EWedge(
    EPoint center,
    EPoint origin,
    Single breadth,
    Boolean direct
)
```

```

void EWedge(
    EPoint center,
    Single diameter,
    Single breadth,
    Single originAngle,
    Single amplitude
)

void EWedge(
    EPoint origin,
    EPoint middle,
    EPoint end,
    Single breadth,
    Boolean fullCircle
)

void EWedge(
    EWedge other
)

```

Parameters

center

Center coordinates of the wedge at its nominal position. The default value is **(0,0)**.

diameter

Nominal diameter of the wedge. The default value is **100**.

breadth

Nominal breadth of the wedge. It must be negative or zero. The default value is **-50**.

originAngle

Origin point coordinates of the wedge.

direct

TRUE (default) means that angles increase anticlockwise in a direct coordinate system and clockwise in an inverse coordinate system.

origin

Origin point coordinates of the wedge.

amplitude

Nominal angular amplitude of the wedge. The default value is **360**.

middle

Middle point coordinates of the wedge.

end

End point coordinates of the wedge.

fullCircle

TRUE (default) in case of a full turn wedge. If **fullCircle** is **FALSE**, **origin** and **end** give the wedge's amplitude.

other

Another EWedge object to be copied in the new EWedge object.

EWedge.FullBreadth

Flag indicating if the EWedge has a full breadth (**breadth = radius**).

[VB6]

FullBreadth As Boolean

read-only

EWedge.FullCircle

Flag indicating if the EWedge is a full circle (**amplitude = 2 PI**).

[VB6]

FullCircle As Boolean

read-only

EWedge.GetCorners

Retrieves the coordinates of each corner of the [EWedge](#).

[VB6]

```
void GetCorners(
    EPoint ar,
    EPoint AAr,
    EPoint aRR,
    EPoint AARR
)
```

Parameters

ar

Coordinates of the inner org corner of the [EWedge](#).

AAr

Coordinates of the inner end corner of the [EWedge](#).

aRR

Coordinates of the outer org corner of the [EWedge](#).

AARR

Coordinates of the outer end corner of the [EWedge](#).

EWedge.GetEdges

Retrieves each edge of the [EWedge](#).

[VB6]

```
void GetEdges (
    ELine a,
    ELine AA,
    ECircle r,
    ECircle RR
)
```

Parameters

a

Org edge of the [EWedge](#).

AA

End edge of the [EWedge](#).

r

Inner edge of the [EWedge](#).

*RR*Outer edge of the [EWedge](#).

EWedge.GetInnerPoint

Returns the coordinates of a particular point specified by its location along the inner circle arc.

```
[VB6]
EPoint GetInnerPoint(
    Single fraction
)
```

Parameters

*fraction*Point location expressed as a fraction of the circle arc (range **[-1, +1]**).

EWedge.GetMidEdges

Retrieves the center coordinates of each edge of the wedge fitting gauge.

```
[VB6]
void GetMidEdges(
    EPoint a,
    EPoint AA,
    EPoint r,
    EPoint RR
)
```

Parameters

*a*Center coordinates of the org edge of the [EWedge](#).*AA*

Center coordinates of the end edge of the [EWedge](#).

r

Center coordinates of the inner edge of the [EWedge](#).

RR

Center coordinates of the outer edge of the [EWedge](#).

EWedge.GetOuterPoint

Returns the coordinates of a particular point specified by its location along the outer circle arc.

[VB6]

```
EPoint GetOuterPoint(  
    Single fraction  
)
```

Parameters

fraction

Point location expressed as a fraction of the circle arc (range **[-1, +1]**).

EWedge.GetPoint

Returns the coordinates of a particular point specified by its location along the wedge perimeter.

[VB6]

```
EPoint GetPoint(  
    Single breadthFraction,  
    Single angleFraction  
)
```

Parameters

breadthFraction

Point location expressed as a fraction of the wedge breadth (range -1, +1).

angleFraction

Point location expressed as a fraction of the wedge amplitude (range -1, +1).

EWedge.InnerApex

Inner apex point coordinates of the [EWedge](#).

[VB6]

InnerApex As EPoint

read-only

EWedge.InnerArcLength

Inner circle arc length of the [EWedge](#).

[VB6]

InnerArcLength As Single

read-only

EWedge.InnerDiameter

Inner diameter of the [EWedge](#).

[VB6]

InnerDiameter As Single

read-only

EWedge.InnerEnd

Inner end point coordinates of the [EWedge](#).

[VB6]

InnerEnd As EPoint

read-only

EWedge.InnerOrg

Inner origin point coordinates of the [EWedge](#).

[VB6]

InnerOrg As EPoint

read-only

EWedge.InnerRadius

Inner radius of the EWedge

[VB6]

InnerRadius As Single

read-only

EWedge.operator=

Copies all the data from another EWedge object into the current EWedge object

[VB6]

```
EWedge operator=(  
    EWedge other  
)
```

Parameters

other

EWedge object to be copied

EWedge.OrgAngle

Angular position from where the EWedge extents.

[VB6]

```
OrgAngle As Single
```

read-only

Remarks

A EWedge is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance. The sign of the rotation angle depends whether the field of view is calibrated or not. * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value. * When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

EWedge.OuterApex

Outer apex point coordinates of the [EWedge](#).

[VB6]

OuterApex As EPoint

read-only

Remarks

A EWedge is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.

EWedge.OuterArcLength

Outer circle arc length of the [EWedge](#).

[VB6]

OuterArcLength As Single

read-only

Remarks

A EWedge is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.

EWedge.OuterDiameter

Outer diameter of the [EWedge](#).

[VB6]

OuterDiameter As Single

read-only

Remarks

A EWedge is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.

EWedge.OuterEnd

Outer end point coordinates of the [EWedge](#).

[VB6]

OuterEnd As EPoint

read-only

Remarks

A EWedge is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.

EWedge.OuterOrg

Outer origin point coordinates of the [EWedge](#).

[VB6]

OuterOrg As EPoint

read-only

Remarks

A EWedge is fully defined knowing its nominal position (its center coordinates), its nominal outer radius (diameter), its breadth (must be negative), the angular position from where it extents, its angular amplitude, and its outline tolerance.

EWedge.OuterRadius

Outer radius of the [EWedge](#).

[VB6]

OuterRadius As Single

read-only

Remarks

A EWedge is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.

EWedge.SetDiameters

Sets the nominal inner/outer diameter and breadth of the [EWedge](#).

[VB6]

```
void SetDiameters(
    Single diameter,
    Single breadth
)
```

Parameters

diameter

Outer diameter of the [EWedge](#). The default value is **100**.

breadth

Breadth of the [EWedge](#). It must be negative or zero. Its default value is **-50**.

Remarks

A EWedge is fully defined knowing its nominal position (given by the coordinates of its center), its nominal outer radius (diameter), its breadth, the angular position from where it extends, its angular amplitude and its outline tolerance. By default, the EWedge diameter value is **100**, which means 100 pixels when the field of view is not calibrated and 100 "units" in case of a calibrated field of view.

EWedge.SetFromCenterAndOrigin

Sets the geometric parameters (nominal position and diameter, breadth, angular position and amplitude) of an EWedge object.

[VB6]

```
void SetFromCenterAndOrigin(
    EPoint center,
    EPoint origin,
    Single breadth,
    Boolean direct
)
```

Parameters

center

Center coordinates of the wedge at its nominal position. The default value is **(0,0)**.

origin

Origin point coordinates of the wedge.

breadth

Nominal breadth of the wedge. It must be negative or zero. The default value is **-50**.

direct

TRUE (default) means that angles increase anticlockwise in a direct coordinate system and clockwise in an inverse coordinate system.

Remarks

In a direct coordinate system, the abscissa extends rightwards and the ordinate extends upwards. The coordinate system is said to be inverse if the abscissa extends rightwards and the ordinate extends downwards.

EWedge.SetFromOriginMiddleEnd

Sets the geometric parameters (nominal position and diameter, breadth, angular position and amplitude) of an EWedge object.

[VB6]

```
void SetFromOriginMiddleEnd(
    EPoint origin,
    EPoint middle,
    EPoint end,
    Single breadth,
    Boolean fullCircle
)
```

Parameters

origin

Origin point coordinates of the wedge.

middle

Middle point coordinates of the wedge.

end

End point coordinates of the wedge.

breadth

Nominal breadth of the wedge. It must be negative or zero. The default value is **-50**.

fullCircle

TRUE (default) in case of a full turn wedge. If **fullCircle** is **FALSE**, **origin** and **end** give the wedge's amplitude.

Remarks

In a direct coordinate system, the abscissa extends rightwards and the ordinate extends upwards. The coordinate system is said to be inverse if the abscissa extends rightwards and the ordinate extends downwards.

EWedge.SetFromTwoPoints

-

```
[VB6]  
void SetFromTwoPoints(  
    EPoint center,  
    EPoint origin,  
    Single breadth,  
    Boolean direct  
)
```

Parameters

center

-

origin

-

breadth

-

direct

-

EWedge.SetRadii

Sets the nominal radius and breadth of the [EWedge](#).

```
[VB6]
```

```
void SetRadii(  
    Single radius,  
    Single breadth  
)
```

Parameters

radius

Outer radius of the [EWedge](#). The default value is **50**.

breadth

Breadth of the EWedge, which must be negative or zero. Its default value is **-50**.

Remarks

A EWedge is fully defined knowing its nominal position (given by the coordinates of its center), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude and its outline tolerance. By default, the EWedge radius value is **50**, which means 50 pixels when the field of view is not calibrated and 50 "units" in case of a calibrated field of view.

EWedgeGauge Class

Manages a wedge fitting gauge.

Base Class: [EWedgeShape](#)

PROPERTIES

Active

Sets the flag indicating whether the gauge is active or not.

ActiveEdges

Active edges as defined in [EDragHandle](#).

AverageDistance	-
FilteringThreshold	-
HVConstraint	-
MeasuredWedge	Information pertaining to the fitted wedge.
MinAmplitude	-
MinArea	-
NumFilteringPasses	-
NumSamples	-
NumSamplesa	Number of sampled points found on edge a during the measure operation.
NumSamplesA	Number of sampled points found on edge A during the measure operation.
NumSamplesr	Number of sampled points found on edge r during the measure operation.

NumSamplesR	Number of sampled points found on edge R during the measure operation.
NumSkipRanges	-
NumValidSamples	-
RectangularSamplingArea	-
SamplingStep	-
Shape	-
Smoothing	-
Thickness	-
Threshold	-
Tolerance	Searching area half thickness of the wedge fitting gauge.
TransitionChoice	-
TransitionIndex	-

TransitionType

-

Type

Shape type.

Valid

-

Wedge

METHODS

Sets the nominal position (center coordinates), diameter, breadth, angular origin and amplitude of the wedge fitting gauge according to a known wedge.

AddSkipRange

Adds an item to the set of skip ranges and returns the index of the newly added range.

CopyTo

Copies all the data of the current EWedgeGauge object into another EWedgeGauge object, and returns it.

Drag

Moves a handle to a new position and updates the position parameters of the gauge.

Draw

Draws a graphical representation of a point location or model fitting gauge, as defined by [EDrawingMode](#).

DrawWithCurrentPen	Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode .
EWedgeGauge	Constructs a wedge measurement context.
GetMeasuredPoint	Returns the coordinates of a sample point, measured along one of the sample paths of the gauge.
GetMinNumFitSamples	Returns the minimum number of samples required for fitting on each side of the shape.
GetSamplea	Allows to retrieve information on the samples found along the a edge.
GetSampleA	Allows to retrieve information on the samples found along the A edge.
GetSampler	Allows to retrieve information on the samples found along the r edge.
GetSampleR	Allows to retrieve information on the samples found along the R edge.
GetSkipRange	Allows to retrieve the start and end values of the skip range corresponding to the given index, if it is valid (i.e. if it has previously been created by a call to the EWedgeGauge::AddSkipRange method).

HitTest	Checks whether the cursor is positioned over a handle (TRUE) or not (FALSE).
Measure	Triggers the point location or the model fitting operation.
MeasureSample	Computes the sample points along the sample path whose index in the list is given by the pathIndex parameter.
MeasureWithoutFitting	Triggers the point location without wedge fitting operation.
operator=	Copies all the data from another EWedgeGauge object into the current EWedgeGauge object
Plot	Draws the profile that is crossed by one of the sample paths of the gauge, as defined by EPlotItem .
PlotWithCurrentPen	Draws the profile that is crossed by one of the sample paths of the gauge, as defined by EPlotItem .
Process	Triggers the process pertaining to a shape or gauge and all the daughter gauges attach to it.

RemoveAllSkipRanges	Removes all the skip ranges previously created by a call to EWedgeGauge::AddSkipRange .
RemoveSkipRange	After a call to EWedgeGauge::AddSkipRange , removes the skip range with the given index.
SetDiameters	Sets the nominal inner/outer diameter and breadth of the EWedgeGauge .
SetFromOriginMiddleEnd	-
SetFromTwoPoints	-
SetMinNumFitSamples	Sets the minimum number of samples required for fitting on each side of the shape.
SetRadii	<p>E Sets the nominal radius and breadth of the EWedgeGauge.</p> <p>W</p>

edgeGauge.Active

Sets the flag indicating whether the gauge is active or not.

[VB6]

Active As Boolean

read-write

Remarks

When complex gauging is required, several gauges can be grouped together. Applying [EWedgeGauge::Process](#) to the mother gauge or shape triggers the measurement of the whole. Only the active gauges will participate in the process. By default, the gauge is active (**TRUE**).

EWedgeGauge.ActiveEdges

Active edges as defined in [EDragHandle](#).

[VB6]

ActiveEdges As Long

read-write

Remarks

In the case of a wedge fitting gauge, each edge can have its own transition detection parameters. Updating the transition parameters only affect the current active edges. By default, all edges are active.

EWedgeGauge.AddSkipRange

Adds an item to the set of skip ranges and returns the index of the newly added range.

[VB6]

```
Long AddSkipRange(
    Long start,
    Long end
)
```

Parameters

start

Beginning of the skip range.

end

End of the skip range.

Remarks

The samples indices between start and end (including the boundaries) will be discarded during the measurement process. The [AddSkipRange](#) method allows to define skip ranges in an [EWedgeGauge](#). This means that, at measure time, samples belonging to these ranges will not be taken into account. A sample may belong to more than one skip range; to be discarded, a sample has to pertain to at least one range. Moreover, the skip ranges are allowed to overlap one another. The range is allowed to be reversed (i.e. end is not required to be greater than start). Also, start and end are not required to reference valid indices at the time of the call (i.e. the range may lie outside of the current return value for [EWedgeGauge::NumSamples](#)).

EWedgeGauge.AverageDistance

-

[VB6]

AverageDistance As Single

read-only

EWedgeGauge.CopyTo

Copies all the data of the current EWedgeGauge object into another EWedgeGauge object, and returns it.

[VB6]

```
EWedgeGauge CopyTo(  
    EWedgeGauge other,  
    Boolean recursive  
)
```

Parameters

other

Pointer to the EWedgeGauge object in which the current EWedgeGauge object data have to be copied.

recursive

TRUE if the children gauges have to be copied as well, **FALSE** otherwise.

Remarks

In case of a **NULL** pointer, a new EWedgeGauge object will be created and returned.

EWedgeGauge.Drag

Moves a handle to a new position and updates the position parameters of the gauge.

```
[VB6]  
void Drag(  
    Long x,  
    Long y  
)
```

Parameters

x

Cursor current coordinates.

y

Cursor current coordinates.

EWedgeGauge.Draw

Draws a graphical representation of a point location or model fitting gauge, as defined by [EDrawingMode](#).

```
[VB6]
```

```
void Draw(
    EDrawAdapter graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)

void Draw(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)

void Draw(
    Long graphicContext,
    ERGBColor color,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by [EDrawingMode](#).

daughters

TRUE if the daughters gauges are to be displayed also.

color

The color in which to draw the overlay.

EWedgeGauge.DrawWithCurrentPen

Draws a graphical representation of a point location or model fitting gauge, as defined by [EDrawingMode](#).

[VB6]

```
void DrawWithCurrentPen(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by [EDrawingMode](#).

daughters

TRUE if the daughters gauges are to be displayed also.

EWedgeGauge.EWedgeGauge

Constructs a wedge measurement context.

[VB6]

```
void EWedgeGauge (
)

void EWedgeGauge (
    EWedgeGauge other
)
```

Parameters

other

Another EWedgeGauge object to be copied in the new EWedgeGauge object.

Remarks

With the default constructor, all the parameters are initialized to their respective default values. With the copy constructor, the constructed wedge measurement context is based on a pre-existing EWedgeGauge object. The gauge children are also copied. Hierarchy copying through a copy constructor is always recursive. To disable this recursion, use instead the [EWedgeGauge::CopyTo](#) method.

EWedgeGauge.FilteringThreshold

-

[VB6]

FilteringThreshold As Single

read-write

EWedgeGauge.GetMeasuredPoint

Returns the coordinates of a sample point, measured along one of the sample paths of the gauge.

[VB6]

```
EPoint GetMeasuredPoint(  
    Long index  
)
```

Parameters

index

This argument must be left unchanged from its default value, i.e. **~0 (= 0xFFFFFFFF)**.

Remarks

These coordinates pertain to the world space; they are expressed in the reference frame to which the current EWedgeGauge object belongs. The gauging process uses a list of sample points to find the shape position and size that best fit a given image. These sample points are measured along the sample paths defined by the gauge geometry.

[EWedgeGauge::GetMeasuredPoint](#) returns the coordinates of the sample point that meets the following two requirements: 1. It lies on the sample path inspected with the last call to [EWedgeGauge::MeasureSample](#), and 1. Among all the sample points along the latter sample path, it is the one selected by the [EWedgeGauge::TransitionChoice](#) property.

Note. For this method to succeed, it is necessary to previously call [EWedgeGauge::MeasureSample](#).

EWedgeGauge.GetMinNumFitSamples

Returns the minimum number of samples required for fitting on each side of the shape.

[VB6]

```
void GetMinNumFitSamples(  
    Long side0,  
    Long side1,  
    Long side2,  
    Long side3  
)
```

Parameters

side0

Minimum number of samples on the top side of the rectangle.

side1

Minimum number of samples on the left side of the rectangle.

side2

Minimum number of samples on the bottom side of the rectangle.

side3

Minimum number of samples on the right side of the rectangle.

Remarks

Irrelevant in case of a point location operation.

EWedgeGauge.GetSampleA

Allows to retrieve information on the samples found along the A edge.

[VB6]

```

Boolean GetSampleA(
    EPoint pt,
    Long index
)

Boolean GetSampleA(
    EPeak pk,
    Long index
)

```

Parameters*pt***EPoint** structure that will receive the sample position.*index*

The sample index

pk

-

Remarks

The method provides the sample point corresponding to the supplied index. The returned value is true when the sample is valid, and false otherwise.

EWedgeGauge.GetSampleA

Allows to retrieve information on the samples found along the A edge.

[VB6]

```

Boolean GetSampleA(
    EPoint pt,
    Long index
)

Boolean GetSampleA(
    EPeak pk,
    Long index
)

```

Parameters*pt*

[EPoint](#) structure that will receive the sample position.

index

The sample index

pk

-

Remarks

The method provides the sample point corresponding to the supplied index. The returned value is true when the sample is valid, and false otherwise.

EWedgeGauge.GetSampleR

Allows to retrieve information on the samples found along the R edge.

[VB6]

```
Boolean GetSampleR(  
    EPoint pt,  
    Long index  
)  
  
Boolean GetSampleR(  
    EPeak pk,  
    Long index  
)
```

Parameters

pt

[EPoint](#) structure that will receive the sample position.

index

The sample index

pk

-

Remarks

The method provides the sample point corresponding to the supplied index. The returned value is true when the sample is valid, and false otherwise.

EWedgeGauge.GetSampleR

Allows to retrieve information on the samples found along the R edge.

```
[VB6]  
  
Boolean GetSampleR(  
    EPoint pt,  
    Long index  
)  
  
Boolean GetSampleR(  
    EPeak pk,  
    Long index  
)
```

Parameters

pt

[EPoint](#) structure that will receive the sample position.

index

The sample index

pk

-

Remarks

The method provides the sample point corresponding to the supplied index. The returned value is true when the sample is valid, and false otherwise.

EWedgeGauge.GetSkipRange

Allows to retrieve the start and end values of the skip range corresponding to the given index, if it is valid (i.e. if it has previously been created by a call to the [EWedgeGauge::AddSkipRange](#) method).

```
[VB6]
```

```
void GetSkipRange(
    Long index,
    Long start,
    Long end
)
```

Parameters*index*

Index of the skip range.

start

Beginning of the skip range.

end

End of the skip range.

Remarks

Start is guaranteed to be smaller or equal to end.

EWedgeGauge.HitTest

Checks whether the cursor is positioned over a handle (**TRUE**) or not (**FALSE**).

[VB6]

```
Boolean HitTest(
    Boolean daughters
)
```

Parameters*daughters*

TRUE if the daughters gauges handles have to be considered as well.

EWedgeGauge.HVConstraint

[VB6]

HVConstraint As Boolean

read-write

EWedgeGauge.Measure

Triggers the point location or the model fitting operation.

[VB6]

```
void Measure(  
    EROIBW8 sourceImage  
)
```

Parameters

sourceImage

Pointer to the source image.

Remarks

When this method is called, and if not enough valid points are detected, then the method returns directly, and the measured gauge is set to the nominal parameters.

EWedgeGauge.MeasuredWedge

Information pertaining to the fitted wedge.

[VB6]

MeasuredWedge As EWedge

read-only

EWedgeGauge.MeasureSample

Computes the sample points along the sample path whose index in the list is given by the **pathIndex** parameter.

[VB6]

```
void MeasureSample(
    EROIBW8 sourceImage,
    Long pathIndex
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

pathIndex

Sample path index.

Remarks

This method stores its results into a temporary variable inside the EWedgeGauge object.

EWedgeGauge.MeasureWithoutFitting

Triggers the point location without wedge fitting operation.

[VB6]

```
void MeasureWithoutFitting(
    EROIBW8 sourceImage
)
```

Parameters

sourceImage

Source image.

Remarks

This method performs the actual measurement for each transition, but does not perform the wedge fitting. This means that individual samples will be available for each edges through the [EWedgeGauge::GetSamplea](#) (Edge a), [EWedgeGauge::GetSampler](#) (Edge r), [EWedgeGauge::GetSampleA](#) (Edge A), [EWedgeGauge::GetSampleR](#) (Edge R) methods, but the gauge position will not be changed. Please note that the filtering will not be performed in this method, since it relies upon the fitting process. The filtering parameters will thus be unused.

EWedgeGauge.MinAmplitude

-

[VB6]

MinAmplitude As Long

read-write

EWedgeGauge.MinArea

-

[VB6]

MinArea As Long

read-write

EWedgeGauge.NumFilteringPasses

-

[VB6]

NumFilteringPasses As Long

read-write

EWedgeGauge.NumSamples

-

[VB6]

NumSamples As Long

read-only

EWedgeGauge.NumSamplesA

Number of sampled points found on edge A during the measure operation.

[VB6]

NumSamplesA As Long

read-only

EWedgeGauge.NumSamplesA

Number of sampled points found on edge A during the measure operation.

[VB6]

NumSamplesA As Long

read-only

EWedgeGauge.NumSamplesR

Number of sampled points found on edge R during the measure operation.

[VB6]

NumSamplesR As Long

read-only

EWedgeGauge.NumSamplesR

Number of sampled points found on edge R during the measure operation.

[VB6]

NumSamplesR As Long

read-only

EWedgeGauge.NumSkipRanges

[VB6]

NumSkipRanges As Long

read-only

EWedgeGauge.NumValidSamples

-

[VB6]

NumValidSamples As Long

read-only

EWedgeGauge.operator=

Copies all the data from another EWedgeGauge object into the current EWedgeGauge object

[VB6]

```
EWedgeGauge operator=(  
    EWedgeGauge other  
)
```

Parameters

other

EWedgeGauge object to be copied

EWedgeGauge.Plot

Draws the profile that is crossed by one of the sample paths of the gauge, as defined by [EPlotItem](#).

[VB6]

```
void Plot(
    EDrawAdapter graphicContext,
    EPlotItem drawItems,
    Single width,
    Single height,
    Single originX,
    Single originY
)

void Plot(
    Long graphicContext,
    EPlotItem drawItems,
    Single width,
    Single height,
    Single originX,
    Single originY
)

void Plot(
    Long graphicContext,
    ERGBColor color,
    EPlotItem drawItems,
    Single width,
    Single height,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawItems

Boolean combination of [EPlotItem](#) members, that indicates which items must be displayed.

width

Width of the plot.

height

Height of the plot.

originX

Origin point coordinates of the plot along the X axis.

originY

Origin point coordinates of the plot along the Y axis.

color

The color in which to draw the overlay.

Remarks

The sample path that is taken into considered is the one inspected with the last call to [EWedgeGauge::MeasureSample](#).

Note. For this method to succeed, it is necessary to previously call [EWedgeGauge::MeasureSample](#).

EWedgeGauge.PlotWithCurrentPen

Draws the profile that is crossed by one of the sample paths of the gauge, as defined by [EPlotItem](#).

[VB6]

```
void PlotWithCurrentPen(
    Long graphicContext,
    EPlotItem drawItems,
    Single width,
    Single height,
    Single originX,
    Single originY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

*drawItems*Boolean combination of [EPlotItem](#) members, that indicates which items must be displayed.*width*

Width of the plot.

height

Height of the plot.

originX

Origin point coordinates of the plot along the X axis.

originY

Origin point coordinates of the plot along the Y axis.

Remarks

The sample path that is taken into considered is the one inspected with the last call to [EWedgeGauge::MeasureSample](#).

Note. For this method to succeed, it is necessary to previously call [EWedgeGauge::MeasureSample](#).

EWedgeGauge.Process

Triggers the process pertaining to a shape or gauge and all the daughter gauges attach to it.

[VB6]

```
void Process(
    EROIBW8 sourceImage,
    Boolean daughters
)
```

Parameters

sourceImage

Pointer to the source image.

daughters

Flag indicating whether the daughters shapes inherit of the same behavior.

Remarks

When complex gauging is required, several gauges can be grouped together. Applying **Process** to the mother gauge or shape triggers the measurement of the whole. Only the active gauges will participate in the process.

EWedgeGauge.RectangularSamplingArea

-

[VB6]

RectangularSamplingArea As Boolean

read-write

EWedgeGauge.RemoveAllSkipRanges

Removes all the skip ranges previously created by a call to [EWedgeGauge::AddSkipRange](#).

[VB6]

```
void RemoveAllSkipRanges()  
)
```

EWedgeGauge.RemoveSkipRange

After a call to [EWedgeGauge::AddSkipRange](#), removes the skip range with the given index.

[VB6]

```
void RemoveSkipRange(  
Long index  
)
```

Parameters

index

Index of the skip range to remove, as returned by [EWedgeGauge::AddSkipRange](#).

EWedgeGauge.SamplingStep

-

[VB6]

SamplingStep As Single

read-write

EWedgeGauge.SetDiameters

Sets the nominal inner/outer diameter and breadth of the [EWedgeGauge](#).

[VB6]

```
void SetDiameters(  
    Single diameter,  
    Single breadth  
)
```

Parameters

diameter

Outer diameter of the [EWedgeGauge](#). The default value is **100**.

breadth

Breadth of the [EWedgeGauge](#). It must be negative or zero. Its default value is **-50**.

Remarks

A EWedgeGauge is fully defined knowing its nominal position (given by the coordinates of its center), its nominal outer radius (diameter), its breadth, the angular position from where it extends, its angular amplitude and its outline tolerance. By default, the EWedgeGauge diameter value is **100**, which means 100 pixels when the field of view is not calibrated and 100 "units" in case of a calibrated field of view.

EWedgeGauge.SetFromOriginMiddleEnd

-

[VB6]

```
void SetFromOriginMiddleEnd(  
    EPoint origin,  
    EPoint middle,  
    EPoint end,  
    Single breadth,  
    Boolean full  
)
```

Parameters

origin

middle

end

breadth

full

EWedgeGauge.SetFromTwoPoints

-

[VB6]

```
void SetFromTwoPoints(
    EPoint center,
    EPoint origin,
    Single breadth,
    Boolean direct
)
```

Parameters*center*

-

origin

-

breadth

-

direct

-

EWedgeGauge.SetMinNumFitSamples

Sets the minimum number of samples required for fitting on each side of the shape.

[VB6]

```
void SetMinNumFitSamples(
    Long side0,
    Long side1,
    Long side2,
    Long side3
)
```

Parameters*side0*

Minimum number of samples on the *outer circle* of the wedge. The default value is **3**.

side1

Minimum number of samples on the *original border* of the wedge. If this value is not specified, it is equal to **n32Side0**. The default value is **2**.

side2

Minimum number of samples on the *inner circle* of the wedge. If this value is not specified, it is equal to **n32Side0**. The default value is **3**.

side3

Minimum number of samples on the *end border* of the wedge. If this value is not specified, it is equal to **n32Side1**. The default value is **2**.

Remarks

Irrelevant in case of a point location operation. When the [EWedgeGauge::Measure](#) method is called, and if not enough valid points are detected, then the method returns directly, and the measured gauge is set to the nominal parameters.

EWedgeGauge.SetRadii

Sets the nominal radius and breadth of the [EWedgeGauge](#).

[VB6]

```
void SetRadii(
    Single outerRadius,
    Single breadth
)
```

Parameters

outerRadius

-

breadth

Breadth of the EWedgeGauge, which must be negative or zero. Its default value is **-50**.

Remarks

A EWedgeGauge is fully defined knowing its nominal position (given by the coordinates of its center), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude and its outline tolerance. By default, the EWedgeGauge radius value is **50**, which means 50 pixels when the field of view is not calibrated and 50 "units" in case of a calibrated field of view.

EWedgeGauge.Shape

[VB6]

Shape As EWedge

read-only

EWedgeGauge.Smoothing

-

[VB6]

Smoothing As Long

read-write

EWedgeGauge.Thickness

-

[VB6]

Thickness As Long

read-write

EWedgeGauge.Threshold

-

[VB6]**Threshold As Long**

read-write

EWedgeGauge.Tolerance

Searching area half thickness of the wedge fitting gauge.

[VB6]**Tolerance As Single**

read-write

Remarks

A wedge fitting gauge is fully defined knowing its nominal position (its center coordinates), its outer nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance. By default, the searching area thickness of the wedge fitting gauge is **20** (2x10), which means 20 pixels when the field of view is not calibrated, and 20 physical units in case of a calibrated field of view.

EWedgeGauge.TransitionChoice

-

[VB6]**TransitionChoice As ETransitionChoice**

read-write

EWedgeGauge.TransitionIndex

-

[VB6]

TransitionIndex As Long

read-write

EWedgeGauge.TransitionType

-

[VB6]

TransitionType As ETransitionType

read-write

EWedgeGauge.Type

Shape type.

[VB6]

Type As EShapeType

read-only

EWedgeGauge.Valid

-

[VB6]

Valid As Boolean

read-only

EWedgeGauge.Wedge

Sets the nominal position (center coordinates), diameter, breadth, angular origin and amplitude of the wedge fitting gauge according to a known wedge.

[VB6]

Wedge As EWedge

read-write

EWedgeShape Class

Base Class: EShape

Derived Class(es): EWedgeGauge

PROPERTIES

Amplitude

Angle	-
ApexAngle	-
Breadth	-
Center	-
CenterX	-
CenterY	-
Direct	-
EndAngle	-
FullBreadth	Flag indicating if the EWedgeShape has a full breadth (breadth = radius).
FullCircle	Flag indicating if the EWedgeShape is a full circle (amplitude = 2 PI).
InnerApex	-
InnerArcLength	-

InnerDiameter	-
InnerEnd	-
InnerOrg	-
InnerRadius	-
OrgAngle	-
OuterApex	-
OuterArcLength	-
OuterDiameter	-
OuterEnd	-
OuterOrg	-
OuterRadius	-
Scale	-

Type	Shape type.
Wedge	M -
THODS	
Closest	Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use EShape::ClosestShape .
CopyTo	-
Drag	-
Draw	Draws a graphical representation of a shape, as defined by EDrawingMode .
DrawWithCurrentPen	Draws a graphical representation of a shape, as defined by EDrawingMode .
EWedgeShape	Constructs a EWedgeShape object.
GetCorners	Retrieves the coordinates of each corner of the EWedgeShape .
GetEdges	Retrieves each edge of the EWedgeShape .

GetInnerPoint	Returns the coordinates of a particular point specified by its location along the inner circle arc.
GetMidEdges	Retrieves the center coordinates of each edge of the wedge fitting gauge.
GetOuterPoint	Returns the coordinates of a particular point specified by its location along the outer circle arc.
GetPoint	Returns the coordinates of a particular point specified by its location along the wedge perimeter.
HitTest	-
operator=	Copies all the data from another EWedgeShape object into the current EWedgeShape object
SetCenterXY	Sets the center coordinates of a EWedgeShape object.
SetDiameters	Sets the nominal inner/outer diameter and breadth of the EWedgeShape .
SetFromCenterAndOrigin	-
SetFromOriginMiddleEnd	-

[SetFromTwoPoints](#)

[SetRadii](#)

E Sets the nominal radius and breadth of the
[EWedgeShape](#).

W

edgeShape.Amplitude

[VB6]

Amplitude As Single

read-write

EWedgeShape.Angle

[VB6]

Angle As Single

read-write

EWedgeShape.ApexAngle

[VB6]

ApexAngle As Single

read-only

EWedgeShape.Breadth

-

[VB6]

Breadth As Single

read-only

EWedgeShape.Center

-

[VB6]

Center As EPoint

read-write

EWedgeShape.CenterX

-

[VB6]

CenterX As Single

read-only

EWedgeShape.CenterY

-

[VB6]

CenterY As Single

read-only

EWedgeShape.Closest

Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use [EShape::ClosestShape](#).

[VB6]

```
void Closest(  
)
```

EWedgeShape.CopyTo

-

[VB6]

```
EWedgeShape CopyTo(  
    EWedgeShape dest,  
    Boolean bRecursive  
)
```

Parameters

dest

-

bRecursive

-

EWedgeShape.Direct

[VB6]

Direct As Boolean

read-write

EWedgeShape.Drag

[VB6]

```
void Drag(  
    Long cursorX,  
    Long cursorY  
)
```

Parameters*cursorX*
-*cursorY*
-

EWedgeShape.Draw

Draws a graphical representation of a shape, as defined by [EDrawingMode](#).

[VB6]

```
void Draw(
    EDrawAdapter graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)

void Draw(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)

void Draw(
    Long graphicContext,
    ERGBColor color,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

Parameters*graphicContext*

Handle of the device context on which to draw.

*drawingMode*Indicates how the point location or model fitting gauge must be displayed, as defined by [EDrawingMode](#).*daughters***TRUE** if the daughters gauges are to be displayed also.*color*
-

EWedgeShape.DrawWithCurrentPen

Draws a graphical representation of a shape, as defined by [EDrawingMode](#).

[VB6]

```
void DrawWithCurrentPen(
    Long graphicContext,
    EDrawingMode drawingMode,
    Boolean daughters
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by [EDrawingMode](#).

daughters

TRUE if the daughters gauges are to be displayed also.

EWedgeShape.EndAngle

-

[VB6]

EndAngle As Single

read-only

EWedgeShape.EWedgeShape

Constructs a EWedgeShape object.

```
[VB6]  
void EWedgeShape()  
)  
  
void EWedgeShape(  
EWedgeShape other  
)
```

Parameters

other

Another EWedgeShape object to be copied in the new EWedgeShape object.

EWedgeShape.FullBreadth

Flag indicating if the EWedgeShape has a full breadth (**breadth = radius**).

```
[VB6]  
FullBreadth As Boolean  
read-only
```

EWedgeShape.FullCircle

Flag indicating if the EWedgeShape is a full circle (**amplitude = 2 PI**).

[VB6]

FullCircle As Boolean

read-only

EWedgeShape.GetCorners

Retrieves the coordinates of each corner of the [EWedgeShape](#).

[VB6]

```
void GetCorners(  
    EPoint ar,  
    EPoint AAr,  
    EPoint aRR,  
    EPoint AARR  
)
```

Parameters

ar

Coordinates of the inner org corner of the [EWedgeShape](#).

AAr

Coordinates of the inner end corner of the [EWedgeShape](#).

aRR

Coordinates of the outer org corner of the [EWedgeShape](#).

AARR

Coordinates of the outer end corner of the [EWedgeShape](#).

EWedgeShape.GetEdges

Retrieves each edge of the [EWedgeShape](#).

[VB6]

```
void GetEdges(
    ELine a,
    ELine AA,
    ECircle r,
    ECircle RR
)
```

Parameters

a

Org edge of the [EWedgeShape](#).

AA

End edge of the [EWedgeShape](#).

r

Inner edge of the [EWedgeShape](#).

RR

Outer edge of the [EWedgeShape](#).

EWedgeShape.GetInnnerPoint

Returns the coordinates of a particular point specified by its location along the inner circle arc.

[VB6]

```
EPoint GetInnerPoint(
    Single fraction
)
```

Parameters

fraction

Point location expressed as a fraction of the circle arc (range **[-1, +1]**).

EWedgeShape.GetMidEdges

Retrieves the center coordinates of each edge of the wedge fitting gauge.

[VB6]

```
void GetMidEdges (
    EPoint a,
    EPoint AA,
    EPoint r,
    EPoint RR
)
```

Parameters

a

Center coordinates of the org edge of the [EWedgeShape](#).

AA

Center coordinates of the end edge of the [EWedgeShape](#).

r

Center coordinates of the inner edge of the [EWedgeShape](#).

RR

Center coordinates of the outer edge of the [EWedgeShape](#).

EWedgeShape.GetOuterPoint

Returns the coordinates of a particular point specified by its location along the outer circle arc.

[VB6]

```
EPoint GetOuterPoint(
    Single fraction
)
```

Parameters

fraction

Point location expressed as a fraction of the circle arc (range **[-1, +1]**).

EWedgeShape.GetPoint

Returns the coordinates of a particular point specified by its location along the wedge perimeter.

[VB6]

```
EPoint GetPoint(  
    Single breadthFraction,  
    Single angleFraction  
)
```

Parameters

breadthFraction

Point location expressed as a fraction of the wedge breadth (range -1, +1).

angleFraction

Point location expressed as a fraction of the wedge amplitude (range -1, +1).

EWedgeShape.HitTest

-

[VB6]

```
Boolean HitTest(  
    Boolean daughters  
)
```

Parameters

daughters

-

EWedgeShape.InnerApex

-

[VB6]

InnerApex As EPoint

read-only

EWedgeShape.InnerArcLength

-

[VB6]

InnerArcLength As Single

read-only

EWedgeShape.InnerDiameter

-

[VB6]

InnerDiameter As Single

read-only

EWedgeShape.InnerEnd

-

[VB6]

InnerEnd As EPoint

read-only

EWedgeShape.InnerOrg

-

[VB6]

InnerOrg As EPoint

read-only

EWedgeShape.InnerRadius

-

[VB6]

InnerRadius As Single

read-only

EWedgeShape.operator=

Copies all the data from another EWedgeShape object into the current EWedgeShape object

[VB6]

```
EWedgeShape operator=(  
    EWedgeShape other  
)
```

Parameters

other

EWedgeShape object to be copied

EWedgeShape.OrgAngle

-

[VB6]

```
OrgAngle As Single  
read-only
```

EWedgeShape.OuterApex

-

[VB6]

OuterApex As EPoint

read-only

EWedgeShape.OuterArcLength

-

[VB6]

OuterArcLength As Single

read-only

EWedgeShape.OuterDiameter

-

[VB6]

OuterDiameter As Single

read-only

EWedgeShape.OuterEnd

-

[VB6]

OuterEnd As EPoint

read-only

EWedgeShape.OuterOrg

-

[VB6]

OuterOrg As EPoint

read-only

EWedgeShape.OuterRadius

-

[VB6]

OuterRadius As Single

read-only

EWedgeShape.Scale

-

[VB6]

Scale As Single

read-write

EWedgeShape.SetCenterXY

Sets the center coordinates of a EWedgeShape object.

[VB6]

```
void SetCenterXY(  
    Single centerX,  
    Single centerY  
)
```

Parameters

centerX

Center coordinates of the EWedgeShape object.

centerY

Center coordinates of the EWedgeShape object.

EWedgeShape.SetDiameters

Sets the nominal inner/outer diameter and breadth of the [EWedgeShape](#).

[VB6]

```
void SetDiameters(  
    Single diameter,  
    Single breadth  
)
```

Parameters

diameter

Outer diameter of the [EWedgeShape](#). The default value is **100**.

breadth

Breadth of the [EWedgeShape](#). It must be negative or zero. Its default value is **-50**.

Remarks

A EWedgeShape is fully defined knowing its nominal position (given by the coordinates of its center), its nominal outer radius (diameter), its breadth, the angular position from where it extents, its angular amplitude and its outline tolerance. By default, the EWedgeShape diameter value is **100**, which means 100 pixels when the field of view is not calibrated and 100 "units" in case of a calibrated field of view.

EWedgeShape.SetFromCenterAndOrigin

-

[VB6]

```
void SetFromCenterAndOrigin(  
    EPoint center,  
    EPoint origin,  
    Single breadth,  
    Boolean direct  
)
```

Parameters

center

-

origin

-

breadth

-

direct

-

EWedgeShape.SetFromOriginMiddleEnd

-

[VB6]

```
void SetFromOriginMiddleEnd(  
    EPoint origin,  
    EPoint middle,  
    EPoint end,  
    Single breadth,  
    Boolean fullCircle  
)
```

Parameters

origin

-

middle

-

end

-

breadth

-

fullCircle

-

EWedgeShape.SetFromTwoPoints

[VB6]

```
void SetFromTwoPoints(  
    EPoint center,  
    EPoint origin,  
    Single breadth,  
    Boolean direct  
)
```

Parameters

center

-

origin

-
breadth

-
direct

EWedgeShape.SetRadii

Sets the nominal radius and breadth of the [EWedgeShape](#).

[VB6]

```
void SetRadii(  
    Single outerRadius,  
    Single breadth  
)
```

Parameters

outerRadius

-
breadth

Breadth of the EWedgeShape, which must be negative or zero. Its default value is **-50**.

Remarks

A EWedgeShape is fully defined knowing its nominal position (given by the coordinates of its center), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude and its outline tolerance. By default, the EWedgeShape radius value is **50**, which means 50 pixels when the field of view is not calibrated and 50 "units" in case of a calibrated field of view.

EWedgeShape.Type

Shape type.

[VB6]

Type As EShapeType

read-only

EWedgeShape.Wedge

-

[VB6]

Wedge As EWedge

read-write

EWorldShape Class

Manages a complete context for calibrating a field of view.

Base Class: [EShape](#)

PROPERTIES

[Angle](#)

Tilt X angle, that is the amplitude of the rotation applied around the X-axis of the sensor to bring the optical (Z) axis perpendicular to the field of view.

[CalibrationModes](#)

Current calibration mode, made from a combination of values.

Center	Horizontal position of the origin point of the reference frame as projected onto the image, i.e. the Sensor abscissa of the point at World coordinates (0,0) .
CenterX	-
CenterY	Vertical position of the origin point of the reference frame as projected onto the image, i.e. the Sensor ordinate of the point at World coordinates (0,0) .
DistortionStrength	Optical distortion strength, i.e. the ratio of the image diagonal length with and without optical distortion introduced by the lens.
DistortionStrength2	Optical distortion strength of the second order.
FieldHeight	Field-of-view height, in physical units.
FieldWidth	Field-of-view width, in physical units.
GridPointsMaxVariation	Maximum variation of grid points (maximum distance between computed grid points and ideal grid points in world space), normalized using the pitch values, after a call to EWorldShape::CalibrationSucceeded

GridPointsMaxVariationThreshold	Grid points maximum variation threshold, that is the value above which the maximum variation of grid points (maximum distance between computed grid points and ideal grid points in world space) is considered too high, and thus marks the calibration as a failure when using EWorldShape::CalibrationSucceeded .
GridPointsMeanVariation	Mean variation of grid points (mean distance between computed grid points and ideal grid points in world space), normalized using the pitch values, after a call to EWorldShape::CalibrationSucceeded .
GridPointsMeanVariationThreshold	Grid points mean variation threshold, that is the value above which the mean variation of grid points (mean distance between computed grid points and ideal grid points in world space) is considered too high, and thus marks the calibration as a failure when using EWorldShape::CalibrationSucceeded .
HitLandmark	-
NumLandmarkElements	-
OpticalCenterX	-
OpticalCenterY	-
PanX	Current horizontal panning factor for drawing operations.

PanY	Current vertical panning factor for drawing operations.
PerspectiveStrength	Perspective effect coefficient, that is the inverse of the observation distance.
Ratio	XResolution/YResolution ratio.
Scale	-
SensorHeight	Logical image height, that is the number of pixels vertically.
SensorWidth	Logical image width, that is the number of pixels horizontally.
TiltXAngle	-
TiltYAngle	Tilt Y angle, that is the amplitude of the rotation applied around the Y-axis of the sensor to bring the optical (Z) axis perpendicular to the field of view.
Type	Shape type.
XResolution	Horizontal sensor resolution, in pixels per unit.

YResolution	Vertical sensor resolution, in pixels per unit.
ZoomX	Current horizontal zooming factor for drawing operations.
ZoomY	M Current vertical zooming factor for drawing operations. E
THODS	
AddLandmark	Adds a new pair of points coordinates (in Sensor and World spaces) to the set of landmarks used for calibration.
AddPoint	Adds a new point coordinates (in Sensor space) to the set of grid points used for calibration.
AutoCalibrate	Returns the best calibration modes for the current calibration grid and calibrates the field of view accordingly.
AutoCalibrateDotGrid	Performs an automatic calibration based on a dot grid image.
AutoCalibrateLandmarks	Returns the best calibration modes for the current landmark set and calibrates the field of view accordingly.

Calibrate	Performs a calibration according to the specified combination of calibration modes.
CalibrationSucceeded	Getter method for the CalibrationSucceeded property. This property is the flag indicating if the calibration has succeeded (TRUE), that is whether the mean variation of grid points (distance between computed grid points and ideal grid points in world space) and the maximum variation of grid points are between given tolerances.
Closest	Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use EShape::ClosestShape .
DisableTypeFilter	-
Drag	-
DragLandmark	-
Draw	Draws the world coordinate axis.
DrawCrossGrid	Draws a regular grid of crosses in world coordinates.
DrawCrossGridWithCurrentPen	Draws a regular grid of crosses in world coordinates.

DrawGrid	Draws the reconstructed grid to be used for grid calibration.
DrawGridWithCurrentPen	Draws the reconstructed grid to be used for grid calibration.
DrawLandmarks	-
DrawWithCurrentPen	Draws the world coordinate axis.
EmptyLandmarks	Resets the landmark specification sequence.
EnableTypeFilter	-
EWorldShape	Constructs a EWorldShape object.
GetLandmarkElement	-
HitLandmarks	Checks if the cursor is placed over a landmark point.
HitTest	-
operator=	Copies all the data from another EWorldShape object into the current EWorldShape object

RebuildGrid	Reconstructs the grid of points from the given dot centers, to compute the World coordinates of the points.
RemoveLandmark	-
SensorToWorld	Performs coordinate transform for arbitrary points from Sensor space to World space.
SetCenterXY	Sets the center coordinates of a EWorldShape object.
SetDistortion	Sets the optical distortion parameters
SetFieldSize	Sets the field of view size in physical units.
SetPan	Sets the horizontal and vertical panning factors for drawing operations.
SetPerspective	Sets the perspective effect coefficient, i.e. the inverse of the observation distance.
SetResolution	Sets the sensor resolution in pixels per unit in both directions.

SetSensor	Initializes the calibration object using all given parameters.
SetSensorSize	Sets the logical image size, i.e. the number of pixels horizontally and vertically.
SetSize	Sets the frame size.
SetupUnwarp	Prepares a lookup table for fast image unwarping.
SetZoom	Sets the horizontal and vertical zooming factors for drawing operations.
Unwarp	Unwarps a distorted image using the current calibration model.
WorldToSensor	Performs coordinate transform for arbitrary points from World space to Sensor space. W

orlShape.AddLandmark

Adds a new pair of points coordinates (in Sensor and World spaces) to the set of landmarks used for calibration.

[VB6]

```
void AddLandmark(  
    EPoint sensorPoint,  
    EPoint worldPoint  
)
```

Parameters

sensorPoint

Sensor point coordinates.

worldPoint

Corresponding World point coordinates.

Remarks

Landmark calibration is the process of computing the calibration parameters by means of a set of known points for which the coordinates are available in both World and Sensor spaces. Usually, such points are chosen as salient features on the part or target in view. They must be such that appropriate image processing techniques allow measuring their positions from the image (directly or indirectly by geometric constructions), while at the same time their coordinates in a reference frame are known.

EWorldShape.AddPoint

Adds a new point coordinates (in Sensor space) to the set of grid points used for calibration.

[VB6]

```
void AddPoint(  
    EPoint sensorPoint  
)
```

Parameters

sensorPoint

Sensor point coordinates.

Remarks

Grid calibration is the process of computing the calibration parameters by means of a set of points known to lie on a rectangular grid. If the grid pitch is known and one of the points is chosen as the origin point, the points can be used as landmarks. By contrast with the landmark calibration functions, the World coordinates of the grid points need not be specified, nor do they have to be given in any specific order. The calibration algorithm is capable of sorting out the points to reconstruct the grid topology. Typically, this function is used in conjunction with blob analysis to extract the dot centers from a grid of dots. Anyway, any other scheme can be used. The grid of points need not be complete, i.e. some of the nodes may be missing, and the points need not completely fill a rectangular area. Landmark calibration is simply achieved by providing a series of point coordinates (in Sensor space only) and then calling the grid reconstruction function followed by the calibration function.

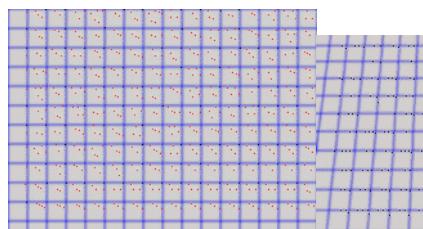
EWorldShape.Angle

Tilt X angle, that is the amplitude of the rotation applied around the X-axis of the sensor to bring the optical (Z) axis perpendicular to the field of view.

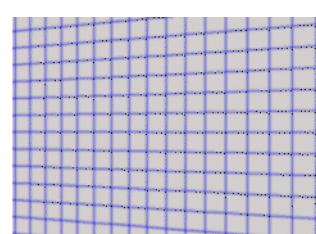
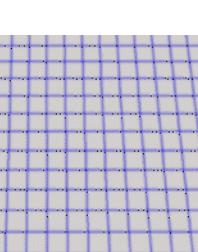
[VB6]

Angle As Single

read-write

Remarks

Tilt Y Angle



Tilt X Angle and

EWorldShape.AutoCalibrate

Returns the best calibration modes for the current calibration grid and calibrates the field of view accordingly.

[VB6]

```
Long AutoCalibrate(  
    Boolean testEmpiricalModes  
)
```

Parameters

testEmpiricalModes

Boolean indicating whether empirical calibration modes ([ECalibrationMode_Quadratic](#) and [ECalibrationMode_Bilinear](#)) should be considered when determining the best calibration modes.

EWorldShape.AutoCalibrateDotGrid

Performs an automatic calibration based on a dot grid image.

[VB6]

```
Long AutoCalibrateDotGrid(  
    EROIBW8 sourceImage,  
    Single columnPitch,  
    Single rowPitch,  
    Boolean testEmpiricalModes  
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

columnPitch

Actual pitches of the grid, i.e. distances between vertical and horizontal rows of the grid.

rowPitch

Actual pitches of the grid, i.e. distances between vertical and horizontal rows of the grid.

testEmpiricalModes

Boolean indicating whether empirical calibration modes ([ECalibrationMode_Quadratic](#) and [ECalibrationMode_Bilinear](#)) should be considered when determining the best calibration modes. Default value is **FALSE**.

Remarks

Returns the best calibration mode for the current dot grid. The [AutoCalibrateDotGrid](#) method will first do an automatic blob analysis in order to extract all dots (all blobs whose area is smaller than 5 pixels will be considered as noise and rejected). The dot gravity centers are used as the grid reference points. Then, the [AutoCalibrateDotGrid](#) method will select and compute the best calibration mode by reducing the fitting error.

EWorldShape.AutoCalibrateLandmarks

Returns the best calibration modes for the current landmark set and calibrates the field of view accordingly.

[VB6]

```
Long AutoCalibrateLandmarks (
    Boolean testEmpiricalModes
)
```

Parameters

testEmpiricalModes

Boolean indicating whether empirical calibration modes ([ECalibrationMode_Quadratic](#) and [ECalibrationMode_Bilinear](#)) should be considered when determining the best calibration modes. Default value is **FALSE**.

Remarks

Landmark calibration is the process of computing the calibration parameters by means of a set of known points for which the coordinates are available in both World and Sensor spaces. Usually, such points are chosen as salient features on the part or target in view. They must be such that appropriate image processing techniques allow measuring their positions from the image (directly or indirectly by geometric constructions), while at the same time their coordinates in a reference frame are known. The [AutoCalibrateLandmarks](#) method is meant to be used with landmark calibration only. To calibrate automatically your field of view using a dot grid, use the [EWorldShape::AutoCalibrate](#) method instead.

EWorldShape.Calibrate

Performs a calibration according to the specified combination of calibration modes.

[VB6]

```
void Calibrate(  
    Long calibrationModes  
)
```

Parameters

calibrationModes

Calibration modes, as defined by a combination of values from [ECalibrationMode](#).

Remarks

Landmark calibration is the process of computing the calibration parameters by means of a set of known points for which the coordinates are available in both World and Sensor spaces. Usually, such points are chosen as salient features on the part or target in view. They must be such that appropriate image processing techniques allow measuring their positions from the image (directly or indirectly by geometric constructions), while at the same time their coordinates in a reference frame are known. In some cases, not all requested calibration modes are honored. After calibration, [EWorldShape::CalibrationModes](#) returns the actual combination of modes in effect.

EWorldShape.CalibrationModes

Current calibration mode, made from a combination of values.

[VB6]

```
CalibrationModes As Long  
read-write
```

Remarks

The supported calibration modes can be set to [ECalibrationMode_Raw](#), meaning that no calibration at all is performed (the World coordinates are pixel indices), or to the logical sum of other values from [ECalibrationMode](#).

EWorldShape.CalibrationSucceeded

Getter method for the **CalibrationSucceeded** property. This property is the flag indicating if the calibration has succeeded (**TRUE**), that is whether the mean variation of grid points (distance between computed grid points and ideal grid points in world space) and the maximum variation of grid points are between given tolerances.

[VB6]

```
Boolean CalibrationSucceeded()
    )
```

Remarks

The mean and maximum grid point variations are normalized using the pitch values. By default, tolerances are set to **0.05** (5 %) for the mean, and **0.1** (10 %) for the maximum grid point variation. You can get and set these tolerances using the [EWorldShape::GridPointsMeanVariationThreshold](#) and [EWorldShape::GridPointsMaxVariationThreshold](#) properties.

EWorldShape.Center

Horizontal position of the origin point of the reference frame as projected onto the image, i.e. the Sensor abscissa of the point at World coordinates **(0,0)**.

[VB6]

Center As EPoint

read-write

EWorldShape.CenterX

-

[VB6]

CenterX As Single

read-only

EWorldShape.CenterY

Vertical position of the origin point of the reference frame as projected onto the image, i.e. the Sensor ordinate of the point at World coordinates **(0,0)**.

[VB6]

CenterY As Single

read-only

EWorldShape.Closest

Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use [EShape::ClosestShape](#).

[VB6]

```
void Closest(  
)
```

EWorldShape.DisableTypeFilter

-

[VB6]

```
void DisableTypeFilter()  
)
```

EWorldShape.DistortionStrength

Optical distortion strength, i.e. the ratio of the image diagonal length with and without optical distortion introduced by the lens.

[VB6]

```
DistortionStrength As Single  
read-only
```

EWorldShape.DistortionStrength2

Optical distortion strength of the second order.

[VB6]

```
DistortionStrength2 As Single  
read-only
```

EWorldShape.Drag

-

[VB6]

```
void Drag(
    Long n32CursorX,
    Long n32CursorY
)
```

Parameters

n32CursorX

n32CursorY

EWorldShape.DragLandmark

-

[VB6]

```
void DragLandmark(
    Long n32CursorX,
    Long n32CursorY
)
```

Parameters

n32CursorX

n32CursorY

EWorldShape.Draw

Draws the world coordinate axis.

```
[VB6]  
  
void Draw(  
    Long graphicContext,  
    EDrawingMode drawingModes,  
    Boolean daughters  
)  
  
void Draw(  
    Long graphicContext,  
    ERGBColor color,  
    EDrawingMode drawingModes,  
    Boolean daughters  
)  
  
void Draw(  
    EDrawAdapter graphicContext,  
    EDrawingMode drawingModes,  
    Boolean daughters  
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingModes

Indicates how the world coordinate axis must be displayed, as defined by [EDrawingMode](#).

daughters

Indicates whether the daughter shapes are to be displayed as well.

color

The color in which to draw the overlay.

EWorldShape.DrawCrossGrid

Draws a regular grid of crosses in world coordinates.

[VB6]

```
void DrawCrossGrid(
    Long graphicContext,
    Single minimumX,
    Single maximumX,
    Single minimumY,
    Single maximumY,
    Long numberOfIntervalsX,
    Long numberOfIntervalsY
)

void DrawCrossGrid(
    Long graphicContext,
    ERGBColor color,
    Single minimumX,
    Single maximumX,
    Single minimumY,
    Single maximumY,
    Long numberOfIntervalsX,
    Long numberOfIntervalsY
)

void DrawCrossGrid(
    EDrawAdapter graphicContext,
    Single minimumX,
    Single maximumX,
    Single minimumY,
    Single maximumY,
    Long numberOfIntervalsX,
    Long numberOfIntervalsY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

minimumX

Abscissa of the leftmost crosses, in world coordinates.

maximumX

Abscissa of the rightmost crosses, in world coordinates.

minimumY

Ordinate of the leftmost crosses, in world coordinates.

maximumY

Ordinate of the rightmost crosses, in world coordinates.

numberOfIntervalsX

Number of intervals between crosses along the horizontal direction.

numberOfIntervalsY

Number of intervals between crosses along the vertical direction.

color

The color in which to draw the overlay.

EWorldShape.DrawCrossGridWithCurrentPen

Draws a regular grid of crosses in world coordinates.

[VB6]

```
void DrawCrossGridWithCurrentPen(
    Long graphicContext,
    Single minimumX,
    Single maximumX,
    Single minimumY,
    Single maximumY,
    Long numberOfIntervalsX,
    Long numberOfIntervalsY
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

minimumX

Abscissa of the leftmost crosses, in world coordinates.

maximumX

Abscissa of the rightmost crosses, in world coordinates.

minimumY

Ordinate of the leftmost crosses, in world coordinates.

maximumY

Ordinate of the rightmost crosses, in world coordinates.

numberOfIntervalsX

Number of intervals between crosses along the horizontal direction.

numberOfIntervalsY

Number of intervals between crosses along the vertical direction.

EWorldShape.DrawGrid

Draws the reconstructed grid to be used for grid calibration.

```
[VB6]  
  
void DrawGrid(  
    Long graphicContext  
)  
  
void DrawGrid(  
    Long graphicContext,  
    ERGBColor color  
)  
  
void DrawGrid(  
    EDrawAdapter graphicContext  
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

color

The color in which to draw the overlay.

EWorldShape.DrawGridWithCurrentPen

Draws the reconstructed grid to be used for grid calibration.

```
[VB6]  
  
void DrawGridWithCurrentPen(  
    Long graphicContext  
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

EWorldShape.DrawLandmarks

-

[VB6]

```
void DrawLandmarks(
    Long graphicContext
)
void DrawLandmarks(
    EDrawAdapter graphicContext
)
```

Parameters

graphicContext

-

EWorldShape.DrawWithCurrentPen

Draws the world coordinate axis.

[VB6]

```
void DrawWithCurrentPen(
    Long graphicContext,
    EDrawingMode drawingModes,
    Boolean daughters
)
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingModes

Indicates how the world coordinate axis must be displayed, as defined by [EDrawingMode](#).

daughters

Indicates whether the daughter shapes are to be displayed as well.

EWorldShape.EmptyLandmarks

Resets the landmark specification sequence.

[VB6]

```
void EmptyLandmarks()  
)
```

Remarks

Landmark calibration is the process of computing the calibration parameters by means of a set of known points for which the coordinates are available in both World and Sensor spaces. Usually, such points are chosen as salient features on the part or target in view. They must be such that appropriate image processing techniques allow measuring their positions from the image (directly or indirectly by geometric constructions), while at the same time their coordinates in a reference frame are known.

EWorldShape.EnableTypeFilter

[VB6]

```
void EnableTypeFilter(  
Long un32Types  
)
```

Parameters

un32Types

EWorldShape.EWorldShape

Constructs a EWorldShape object.

```
[VB6]  
void EWorldShape(  
    EWorldShape other  
)  
  
void EWorldShape(  
)
```

Parameters

other

Another EWorldShape object to be copied in the new EWorldShape object.

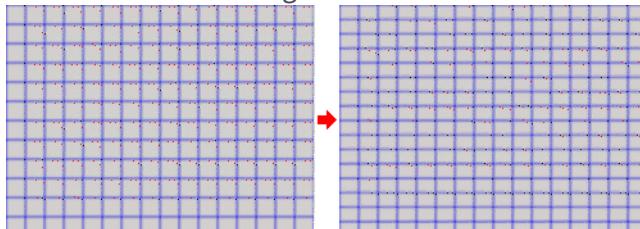
EWorldShape.FieldHeight

Field-of-view height, in physical units.

```
[VB6]  
FieldHeight As Single  
read-only
```

Remarks

Field size not matching the sensor size results in non-square pixels.



Pixels having non-square aspect

ratioBeware there is a restriction pertaining to the allowed image anisotropy. The resulting pixels aspect ratio (**XResolution/YResolution**) should be in the range **[-4/3, -3/4]** (or **[3/4, 4/3]**), otherwise the calibration process could fail.

EWorldShape.FieldWidth

Field-of-view width, in physical units.

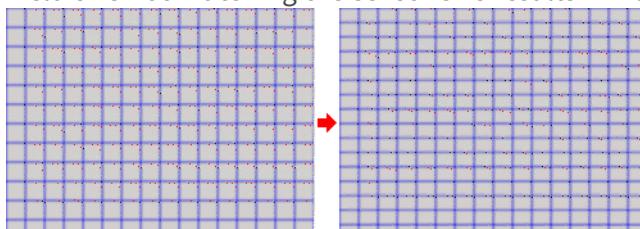
[VB6]

FieldWidth As Single

read-only

Remarks

Field size not matching the sensor size results in non-square pixels.



Pixels having non-square aspect

ratioBeware there is a restriction pertaining to the allowed image anisotropy. The resulting pixels aspect ratio (**XResolution/YResolution**) should be in the range **[-4/3, -3/4]** (or **[3/4, 4/3]**), otherwise the calibration process could fail.

EWorldShape.GetLandmarkElement

[VB6]

```
ELandmark GetLandmarkElement(  
    Long i  
)  
  
ELandmark GetLandmarkElement(  
    Long i  
)
```

Parameters

i

-

EWorldShape.GridPointsMaxVariation

Maximum variation of grid points (maximum distance between computed grid points and ideal grid points in world space), normalized using the pitch values, after a call to [EWorldShape::CalibrationSucceeded](#)

[VB6]

```
GridPointsMaxVariation As Single  
read-only
```

Remarks

The maximum grid point variation is normalized using the pitch values.

EWorldShape.GridPointsMaxVariationThreshold

Grid points maximum variation threshold, that is the value above which the maximum variation of grid points (maximum distance between computed grid points and ideal grid points in world space) is considered too high, and thus marks the calibration as a failure when using [EWorldShape::CalibrationSucceeded](#).

[VB6]

GridPointsMaxVariationThreshold As Single

read-write

Remarks

The maximum grid point variation is normalized using the pitch values.

EWorldShape.GridPointsMeanVariation

Mean variation of grid points (mean distance between computed grid points and ideal grid points in world space), normalized using the pitch values, after a call to [EWorldShape::CalibrationSucceeded](#).

[VB6]

GridPointsMeanVariation As Single

read-only

Remarks

The mean grid point variation is normalized using the pitch values.

EWorldShape.GridPointsMeanVariationThreshold

Grid points mean variation threshold, that is the value above which the mean variation of grid points (mean distance between computed grid points and ideal grid points in world space) is considered too high, and thus marks the calibration as a failure when using [EWorldShape::CalibrationSucceeded](#).

[VB6]

GridPointsMeanVariationThreshold As Single

read-write

Remarks

The mean grid point variation is normalized using the pitch values.

EWorldShape.HitLandmark

-

[VB6]

HitLandmark As Long

read-only

EWorldShape.HitLandmarks

Checks if the cursor is placed over a landmark point.

[VB6]

```
void HitLandmarks()  
)
```

Remarks

Landmark calibration is the process of computing the calibration parameters by means of a set of known points for which the coordinates are available in both World and Sensor spaces. Usually, such points are chosen as salient features on the part or target in view. They must be such that appropriate image processing techniques allow measuring their positions from the image (directly or indirectly by geometric constructions), while at the same time their coordinates in a reference frame are known.

EWorldShape.HitTest

-

[VB6]

```
Boolean HitTest(  
    Boolean bDaughters  
)
```

Parameters

bDaughters

-

EWorldShape.NumLandmarkElements

-

[VB6]

NumLandmarkElements As Long
read-only

EWorldShape.operator=

Copies all the data from another EWorldShape object into the current EWorldShape object

[VB6]

```
EWorldShape operator=(
    EWorldShape other
)
```

Parameters

other

EWorldShape object to be copied

EWorldShape.OpticalCenterX

-

```
[VB6]
OpticalCenterX As Single
read-only
```

EWorldShape.OpticalCenterY

-

```
[VB6]
OpticalCenterY As Single
read-only
```

EWorldShape.PanX

Current horizontal panning factor for drawing operations.

[VB6]

PanX As Single

read-only

EWorldShape.PanY

Current vertical panning factor for drawing operations.

[VB6]

PanY As Single

read-only

EWorldShape.PerspectiveStrength

Perspective effect coefficient, that is the inverse of the observation distance.

[VB6]

PerspectiveStrength As Single

read-only

Remarks

The larger this parameter, the more perceivable the perspective distortion will be. A **NULL** value corresponds to a telecentric lens.

EWorldShape.Ratio

XResolution/YResolution ratio.

[VB6]

Ratio As Single

read-write

Remarks

If **Ratio** equals **-1** (or **1**), pixels are square. Beware there is a restriction pertaining to the allowed image anisotropy. The resulting pixels aspect ratio (**XResolution/YResolution**) should be in the range **[-4/3, -3/4]** (or **[3/4, 4/3]**), otherwise the calibration process could fail.

EWorldShape.RebuildGrid

Reconstructs the grid of points from the given dot centers, to compute the World coordinates of the points.

[VB6]

```
Long RebuildGrid(
    Single colPitch,
    Single rowPitch,
    Long centerIndex,
    Boolean direct
)

Long RebuildGrid(
    Single colPitch,
    Single rowPitch,
    EPoint worldCenter,
    Long centerIndex,
    Boolean direct
)
```

Parameters

colPitch

Actual pitches of the grid, that are distances between vertical and horizontal rows of the grid.

rowPitch

Actual pitches of the grid, that are distances between vertical and horizontal rows of the grid.

centerIndex

Index of the grid point chosen as coordinate origin point. By default, the most central grid point.

direct

TRUE if the world reference frame points upwards.

worldCenter

World coordinates of the starting grid point.

Remarks

This member function also returns the number of grid points that were connected. This prepares the calibration using landmarks (for use by member [EWorldShape::Calibrate](#)). Landmark calibration is the process of computing the calibration parameters by means of a set of known points for which the coordinates are available in both World and Sensor spaces. Usually, such points are chosen as salient features on the part or target in view. They must be such that appropriate image processing techniques allow measuring their positions from the image (directly or indirectly by geometric constructions), while at the same time their coordinates in a reference frame are known. See also Dot-Grid-Based Calibration for the grid construction algorithm.

EWorldShape.RemoveLandmark

-

[VB6]

```
void RemoveLandmark(
    Long index
)
```

Parameters

index

-

EWorldShape.Scale

-

[VB6]

Scale As Single

read-write

EWorldShape.SensorHeight

Logical image height, that is the number of pixels vertically.

[VB6]

SensorHeight As Long

read-only

EWorldShape.SensorToWorld

Performs coordinate transform for arbitrary points from Sensor space to World space.

[VB6]

```
EPoint SensorToWorld(  
    EPoint sensorPoint  
)
```

Parameters

sensorPoint

Sensor point.

EWorldShape.SensorWidth

Logical image width, that is the number of pixels horizontally.

[VB6]

SensorWidth As Long

read-only

EWorldShape.SetCenterXY

Sets the center coordinates of a EWorldShape object.

[VB6]

```
void SetCenterXY(  
    Single centerX,  
    Single centerY  
)
```

Parameters

centerX

Center coordinates of the EWorldShape object.

centerY

Center coordinates of the EWorldShape object.

EWorldShape.SetDistortion

Sets the optical distortion parameters

[VB6]

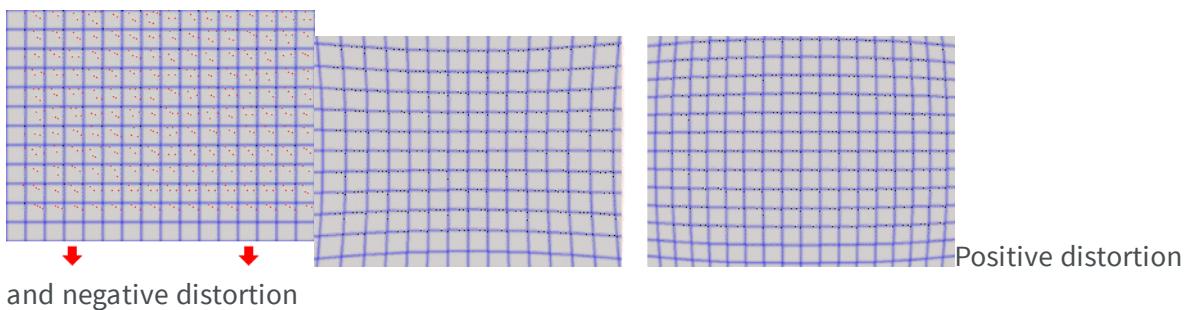
```
void SetDistortion(  
    Single distortionStrength,  
    Single distortionStrength2  
)
```

Parameters*distortionStrength*

Optical distortion strength, i.e. the ratio of the image diagonal length with and without optical distortion introduced by the lens.

distortionStrength2

Optical distortion strength of the second order.

Remarks

EWorldShape.SetFieldSize

Sets the field of view size in physical units.

[VB6]

```
void SetFieldSize(
    Single width,
    Single height
)
```

Parameters*width*

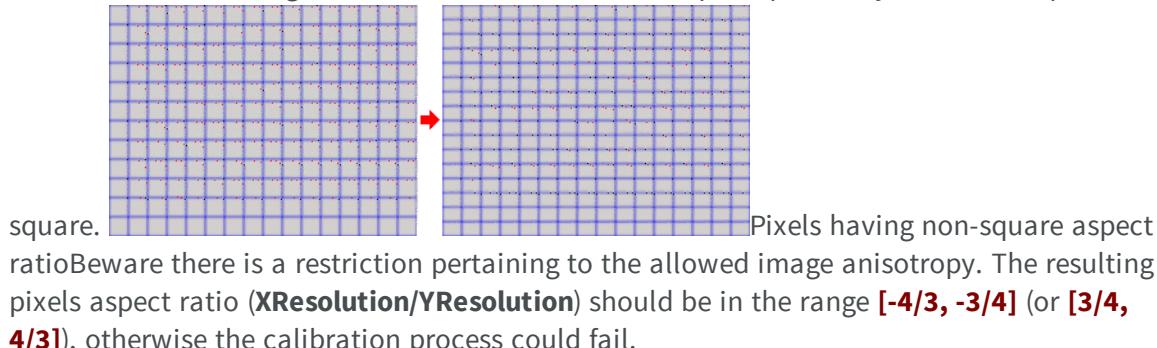
Full image physical width, in length units.

height

Full image physical height, in length units. If not specified, same as physical width.

Remarks

Field size not matching the sensor size results in non-square pixels. By default, the pixels are



EWorldShape.SetPan

Sets the horizontal and vertical panning factors for drawing operations.

[VB6]

```
void SetPan(
    Single panX,
    Single panY
)
```

Parameters

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

Remarks

All objects attached to an **EWorldShape** object inherit of the same panning factor.

EWorldShape.SetPerspective

Sets the perspective effect coefficient, i.e. the inverse of the observation distance.

[VB6]

```
void SetPerspective(
    Single tiltXAngle,
    Single tiltYAngle,
    Single perspectiveStrength
)
```

Parameters

tiltXAngle

Tilt angles, i.e. the amplitudes of the rotations applied around the X and Y axis of the sensor to bring the optical (Z) axis perpendicular to the field of view.

tiltYAngle

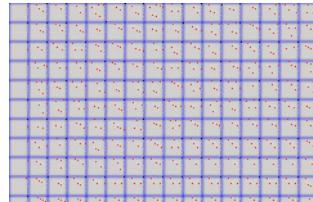
Tilt angles, i.e. the amplitudes of the rotations applied around the X and Y axis of the sensor to bring the optical (Z) axis perpendicular to the field of view.

perspectiveStrength

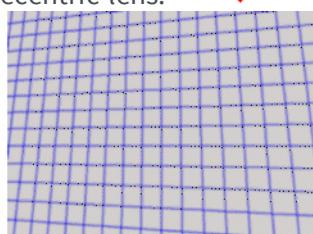
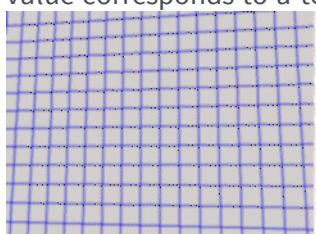
Perspective effect coefficient.

Remarks

The larger this parameter, the more perceivable the perspective distortion will be. A **NULL**



value corresponds to a telecentric lens.



Weak Perspective and Strong Perspective

EWorldShape.SetResolution

Sets the sensor resolution in pixels per unit in both directions.

[VB6]

```
void SetResolution(  
    Single resolutionX,  
    Single resolutionY  
)
```

Parameters

resolutionX

Horizontal resolution in pixels per units

resolutionY

Vertical resolution in pixels per units. If not specified, same as horizontal resolution.

Remarks

By default, the pixels are square.

EWorldShape.SetSensor

Initializes the calibration object using all given parameters.

[VB6]

```
void SetSensor(  
    Long sensorWidth,  
    Long sensorHeight,  
    Single fieldWidth,  
    Single fieldHeight,  
    Single centerX,  
    Single centerY,  
    Single angle,  
    Single tiltXAngle,  
    Single tiltYAngle,  
    Single perspectiveStrength,  
    Single distortionStrength,  
    Single distortionStrength2,  
    Single opticalCenterX,  
    Single opticalCenterY,  
    Long calibrationModes  
)
```

Parameters*sensorWidth*

Logical size of the field of view, i.e. image size, in pixels.

sensorHeight

Logical size of the field of view, i.e. image size, in pixels.

fieldWidth

Physical size of the field of view. By default (argument omitted), the pixels are square.

fieldHeight

Physical size of the field of view. By default (argument omitted), the pixels are square.

centerX

Position of the "intersection" between the optical axis and the field of view in the image.

By default, if the calibration modes contain [ECalibrationMode_Raw](#), it is set to 0.

Otherwise, it is set to the image center.

centerY

Position of the "intersection" between the optical axis and the field of view in the image.

By default, if the calibration modes contain [ECalibrationMode_Raw](#), it is set to 0 (or to the bottommost pixel index if the calibration modes also contain [ECalibrationMode_Inverse](#)).

Otherwise, it is set to the image center.

angle

Skew angle, i.e. angle formed by the axis of reference and the image edges. By default (argument omitted), no skewing effect is assumed.

tiltXAngle

Rotation angles on the X axis to bring the optical axis perpendicular to the image plane. By default (argument omitted), no perspective effect is assumed.

tiltYAngle

Rotation angles on the Y axis to bring the optical axis perpendicular to the image plane. By default (argument omitted), no perspective effect is assumed.

perspectiveStrength

Relative importance of the perspective effect. By default, no perspective effect is assumed, as if the lens was telecentric.

distortionStrength

Relative importance of the lens radial distortion. Positive for barrel, negative for cushion.

By default (argument omitted), no optical distortion is assumed.

distortionStrength2

Relative importance of the lens radial distortion of the second order. By default (argument omitted), no optical distortion of second order is assumed.

opticalCenterX

X Position of the "intersection" between the optical axis and the field of view in the image.

By default (argument omitted) the image center.

opticalCenterY

Y Position of the "intersection" between the optical axis and the field of view in the image.
By default (argument omitted) the image center.

calibrationModes

Desired calibration mode effects to be combined, as defined by [ECalibrationMode](#). By default (argument omitted), the simplest model compatible with the given parameters is chosen.

Remarks

The function automatically selects the appropriate calibration model by checking the parameters. The use of a more complex calibration mode can be enforced by means of parameter [EWorldShape::CalibrationModes](#), not a simpler one.

EWorldShape.SetSensorSize

Sets the logical image size, i.e. the number of pixels horizontally and vertically.

[VB6]

```
void SetSensorSize(  
    Long width,  
    Long height  
)
```

Parameters

width

Full image logical sizes, in pixels.

height

Full image logical sizes, in pixels.

EWorldShape.GetSize

Sets the frame size.

[VB6]

```
void SetSize(
    Single sizeX,
    Single sizeY
)
```

Parameters*sizeX*Frame X-axis length. The default value is **100**.*sizeY*

Frame Y-axis length. By default, both axes have the same length.

Remarks

By default, both frame axis value are set to **100**, which means 100 pixels when the field of view is not calibrated and 100 "units" in case of a calibrated field of view.

EWorldShape.SetupUnwarp

Prepares a lookup table for fast image unwarping.

[VB6]

```
void SetupUnwarp(
    EUnwarpingLut lookupTable,
    EROIBW8 sourceImage,
    Boolean interpolate
)

void SetupUnwarp(
    EUnwarpingLut lookupTable,
    EROIC24 sourceImage,
    Boolean interpolate
)
```

Parameters*lookupTable*

Pointer to the lookup table.

sourceImage

Pointer to the source image/ROI.

interpolate

Interpolation mode. Default value is **FALSE**.

Remarks

The function should be called each time the system is re-calibrated (after the optical setup has been changed, for instance). A sample source image has to be supplied to [SetupUnwarp](#), and its row pitch is recorded in order to speedup the unwarping process. This implies that the following calls to [EWorldShape::Unwarp](#) are not allowed to use images with row pitches different from the source image initially supplied to [SetupUnwarp](#).

EWorldShape.SetZoom

Sets the horizontal and vertical zooming factors for drawing operations.

[VB6]

```
void SetZoom(  
    Single zoomX,  
    Single zoomY  
)
```

Parameters

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to **0**, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

Remarks

All objects attached to an [EWorldShape](#) inherit of the same zooming factor.

EWorldShape.TiltXAngle

[VB6]

TiltXAngle As Single

read-only

EWorldShape.TiltYAngle

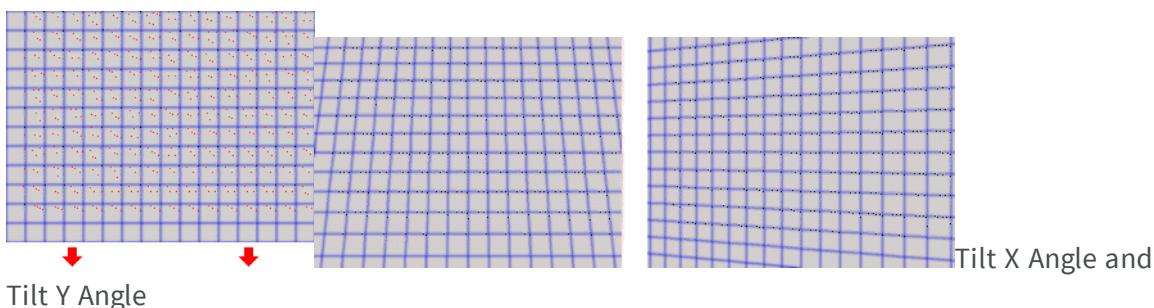
Tilt Y angle, that is the amplitude of the rotation applied around the Y-axis of the sensor to bring the optical (Z) axis perpendicular to the field of view.

[VB6]

TiltYAngle As Single

read-only

Remarks



EWorldShape.Type

Shape type.

[VB6]

Type As EShapeType

read-only

EWorldShape.Unwarp

Unwarps a distorted image using the current calibration model.

```
[VB6]

void Unwarp(
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Boolean interpolate
)

void Unwarp(
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Boolean interpolate
)

void Unwarp(
    EUncappingLut lookupTable,
    EROIBW8 sourceImage,
    EROIBW8 destinationImage,
    Boolean interpolate
)

void Unwarp(
    EUncappingLut lookupTable,
    EROIC24 sourceImage,
    EROIC24 destinationImage,
    Boolean interpolate
)
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination unwarped image.

interpolate

Interpolation mode. Default value is **FALSE**.

lookupTable

Pointer to the lookup table.

Remarks

Using a pre-computed lookup table allows speeding up the unwarping process. The lookup table is initialized by means of the [EWorldShape::SetupUnwarp](#) function.

EWorldShape.WorldToSensor

Performs coordinate transform for arbitrary points from World space to Sensor space.

[VB6]

```
EPoint WorldToSensor(  
    EPoint worldPoint  
)
```

Parameters

worldPoint

EWorldShape.XResolution

Horizontal sensor resolution, in pixels per unit.

[VB6]

```
XResolution As Single  
read-only
```

EWorldShape.YResolution

Vertical sensor resolution, in pixels per unit.

[VB6]

YResolution As Single

read-only

EWorldShape.ZoomX

Current horizontal zooming factor for drawing operations.

[VB6]

ZoomX As Single

read-only

EWorldShape.ZoomY

Current vertical zooming factor for drawing operations.

[VB6]

ZoomY As Single

read-only

Structures

E3DPoint Struct

Represents a 3D point.

PROPERTIES

X	The X coordinate of E3DPoint.
Y	The Y coordinate of E3DPoint.
Z	The Z coordinate of E3DPoint.

METHODS

<code>E3DPoint</code>	Constructs a default E3DPoint object (initialized to 0).
<code>operator!=</code>	Check if two E3DPoint are strictly different
<code>operator==</code>	Check if two E3DPoint are strictly equal

E3DPoint.E3DPoint

Constructs a default E3DPoint object (initialized to 0).

```
[VB6]
void E3DPoint(
    )
void E3DPoint(
    Single x,
    Single y,
    Single z
)
```

Parameters

x

-

y

-

z

-

E3DPoint.operator!=

Check if two E3DPoint are strictly different

```
[VB6]
Boolean operator!=(
    E3DPoint point
)
```

Parameters

point

The other point.

E3DPoint.operator==

Check if two E3DPoint are strictly equal

[VB6]

```
Boolean operator==(  
    E3DPoint point  
)
```

Parameters

point

The other point.

E3DPoint.X

The X coordinate of E3DPoint.

[VB6]

```
Single X
```

E3DPoint.Y

The Y coordinate of E3DPoint.

[VB6]

```
Single Y
```

E3DPoint.Z

The Z coordinate of E3DPoint.

[VB6]

Single Z

EBW1 Struct

Black and white pixel value, coded as an unsigned 32-bit integer.

Remarks

Every pixel is coded on 1 bit, allowing to represent 2 different values. The value **0** stands for black (background), and the value **1** stands for white (foreground).

PROPERTIES

Size	The number of bits in a pixel
UINT32Value	-
Value	The value of the pixel.

METHODS

EBW1	Constructs a EBW1 object.
------	---------------------------

EBW1.EBW1

Constructs a EBW1 object.

[VB6]

```
void EBW1(
)
void EBW1(
    Long value
)
```

Parameters

value

The value of the pixel.

EBW1.Size

The number of bits in a pixel

[VB6]

Size As Long

read-only

EBW1.UINT32Value

-

[VB6]

UINT32Value As Long

read-write

EBW1.Value

The value of the pixel.

[VB6]

Long Value

EBW16 Struct

Gray-level pixel value, coded as an unsigned 16-bit integer.

Remarks

High-quality cameras or scanners are able to digitize on 10 or 12 bits. Sometimes too, to avoid numerical truncation errors, intermediate processing results require more than 8 bits of storage. In such situations, 8 bits gray-level images are no longer sufficient. 16 bits gray-level images are similar to 8 bits ones, but each pixel is, in this case, coded on 16 bits, which effect is to increase the levels of gray to 65,536. It is not possible to show the difference between a gray-level image quantized on 16 bits rather than 8. Under Windows, no display device is able to display 16-bit gray levels. Windows doesn't allow you to display more than 256 gray levels.

PROPERTIES

Size	The number of bits in a pixel
UINT32Value	-
Value	The value of the pixel.

METHODS

EBW16	Constructs a EBW16 object.
-------	----------------------------

EBW16.EBW16

Constructs a EBW16 object.

```
[VB6]  
void EBW16()  
)  
  
void EBW16(  
Long value  
)
```

Parameters

value

The value of the pixel.

EBW16.Size

The number of bits in a pixel

```
[VB6]  
Size As Long  
read-only
```

EBW16.UINT32Value

[VB6]

UINT32Value As Long

read-write

EBW16.Value

The value of the pixel.

[VB6]

Long Value

EBW16Path Struct

Path from a [EBW16](#) image: image pixel coordinates, and associated gray-level pixel value.

PROPERTIES

Pixel	The value of the pixel at this coordinate.
X	Coordinate along the horizontal direction.
Y	Coordinate along the vertical direction.

EBW16Path.Pixel

The value of the pixel at this coordinate.

[VB6]

EBW16 Pixel

EBW16Path.X

Coordinate along the horizontal direction.

[VB6]

Long X

EBW16Path.Y

Coordinate along the vertical direction.

[VB6]

Long Y

EBW32 Struct

Gray-level pixel value, coded as an unsigned 32-bit integer.

PROPERTIES

Size	The number of bits in a pixel
UINT32Value	-

Value	The value of the pixel.
-------	-------------------------

METHODS

EBW32	Constructs a EBW32 object.
-------	----------------------------

EBW32.EBW32

Constructs a EBW32 object.

```
[VB6]  
void EBW32()  
)  
  
void EBW32(  
Long value  
)
```

Parameters

value

The value of the pixel.

EBW32.Size

The number of bits in a pixel

```
[VB6]  
Size As Long  
read-only
```

EBW32.UINT32Value

-

[VB6]

UINT32Value As Long

read-write

EBW32.Value

The value of the pixel.

[VB6]

Long Value

EBW8 Struct

Gray-level pixel value, coded as an unsigned 8-bit integer.

Remarks

Every pixel is coded on 8 bits, allowing to represent 256 different values. The value **0** stands for black (background) and the value **255** stands for white (foreground). The 254 remaining values stand for shades of gray. This is sufficient for most applications. Most of the Open eVision gray-level operations apply to this pixel type.

PROPERTIES

Size	The number of bits in a pixel
------	-------------------------------

UINT32Value	-
Value	The value of the pixel.

METHODS

EBW8	Constructs a EBW8 object.
------	---------------------------

EBW8.EBW8

Constructs a EBW8 object.

```
[VB6]
void EBW8(
)
void EBW8(
    Byte value
)
```

Parameters

value

The value of the pixel.

EBW8.Size

The number of bits in a pixel

```
[VB6]
Size As Long
```

read-only

EBW8.UINT32Value

-

[VB6]

UINT32Value As Long

read-write

EBW8.Value

The value of the pixel.

[VB6]

Byte Value

EBW8Path Struct

Path from a **EBW8** image: image pixel coordinates, and associated gray-level pixel value.

PROPERTIES

Pixel	The value of the pixel at this coordinate.
X	Coordinate along the horizontal direction.

Y	Coordinate along the vertical direction.
---	--

EBW8Path.Pixel

The value of the pixel at this coordinate.

[VB6]
EBW8 Pixel

EBW8Path.X

Coordinate along the horizontal direction.

[VB6]
Long X

EBW8Path.Y

Coordinate along the vertical direction.

[VB6]
Long Y

EC15 Struct

Color pixel value, coded as 3 fields of 5 bits each (red, green, blue components) and 1 field of 1 bit for padding.

Remarks

This class is suited to handle the Windows RGB15 color images. The pixel values are coded on 15 bits, leaving 32 possible levels per color component (red, green or blue).

PROPERTIES

<code>C0</code>	The value of the first component of the pixel (red channel).
<code>C1</code>	The value of the second component of the pixel (green channel).
<code>C2</code>	The value of the third component of the pixel (blue channel).
<code>Size</code>	The number of bits in a pixel
<code>UINT32Value</code>	-

METHODS

<code>EC15</code>	Constructs a EC15 object.
-------------------	---------------------------

EC15.C0

The value of the first component of the pixel (red channel).

[VB6]

Long C0

EC15.C1

The value of the second component of the pixel (green channel).

[VB6]

Long C1

EC15.C2

The value of the third component of the pixel (blue channel).

[VB6]

Long C2

EC15.EC15

Constructs a EC15 object.

[VB6]

```
void EC15(  
    )
```

```
void EC15(
    Byte c0,
    Byte c1,
    Byte c2
)
```

Parameters

c0

The value of the first component of the pixel (red channel).

c1

The value of the second component of the pixel (green channel).

c2

The value of the third component of the pixel (blue channel).

EC15.Size

The number of bits in a pixel

[VB6]

Size As Long

read-only

EC15.UINT32Value

[VB6]

UINT32Value As Long

read-write

EC16 Struct

Color pixel value, coded as 3 fields of 5 bits, 6 bits and 5 bits (red, green and blue components).

Remarks

This class is suited to handle the Windows RGB16 color images. The pixel values are coded on 16 bits (5-6-5), leaving 32 possible levels for R and B components, and 64 possible levels for G component.

PROPERTIES

C0	The value of the first component of the pixel (red channel).
C1	The value of the second component of the pixel (green channel).
C2	The value of the third component of the pixel (blue channel).
Size	The number of bits in a pixel
UINT32Value	-

METHODS

EC16	Constructs a EC16 object.
------	---------------------------

EC16.C0

The value of the first component of the pixel (red channel).

[VB6]

Long C0

EC16.C1

The value of the second component of the pixel (green channel).

[VB6]

Long C1

EC16.C2

The value of the third component of the pixel (blue channel).

[VB6]

Long C2

EC16.EC16

Constructs a EC16 object.

[VB6]

```
void EC16(  
    )
```

```
void EC16(
    Byte c0,
    Byte c1,
    Byte c2
)
```

Parameters

c0

The value of the first component of the pixel (red channel).

c1

The value of the second component of the pixel (green channel).

c2

The value of the third component of the pixel (blue channel).

EC16.Size

The number of bits in a pixel

[VB6]

Size As Long

read-only

EC16.UINT32Value

[VB6]

UINT32Value As Long

read-write

EC24 Struct

Color pixel value coded as 3 unsigned 8-bit integers (red, green and blue components).

Remarks

(RGB triplet, windows 24 bpp bitmap format) The pixel values are coded on 24 bits, providing 256 possible levels per color component. This way, RGB images can represent 16,777,216 different colors. This is sufficient for most applications. Most of the Open eVision color operations apply to this pixel type.

PROPERTIES

C0	The value of the first component of the pixel (red channel).
C1	The value of the second component of the pixel (green channel).
C2	The value of the third component of the pixel (blue channel).
Size	The number of bits in a pixel
UINT32Value	-

METHODS

EC24	Constructs a EC24 object.
------	---------------------------

EC24.C0

The value of the first component of the pixel (red channel).

[VB6]

Byte C0

EC24.C1

The value of the second component of the pixel (green channel).

[VB6]

Byte C1

EC24.C2

The value of the third component of the pixel (blue channel).

[VB6]

Byte C2

EC24.EC24

Constructs a EC24 object.

[VB6]

```
void EC24(  
    )
```

```
void EC24(
    ERGBColor rgbcColor
)

void EC24(
    Byte c0,
    Byte c1,
    Byte c2
)
```

Parameters

rgbcColor

-

c0

The value of the first component of the pixel (red channel).

c1

The value of the second component of the pixel (green channel).

c2

The value of the third component of the pixel (blue channel).

EC24.Size

The number of bits in a pixel

[VB6]

Size As Long

read-only

EC24.UINT32Value

[VB6]

UINT32Value As Long

read-write

EC24A Struct

Color pixel value coded as 4 unsigned 8-bit integers (red, green, blue and alpha components).

Remarks

This class is suited to handle the Windows RGB32 color format. The pixel values are coded on 32 bits, leaving 256 possible levels per color component (red, green or blue), and 8 more bits for an alpha channel. Currently, the alpha channel is not used for any purpose in the Open eVision processing functions. Users are free to use it to store an additional gray-level content.

PROPERTIES

A	The value of the alpha component of the pixel.
C0	The value of the first component of the pixel (red channel).
C1	The value of the second component of the pixel (green channel).
C2	The value of the third component of the pixel (blue channel).
Size	The number of bits in a pixel
UINT32Value	-

METHODS

EC24A	Constructs a EC24A object.
-------	----------------------------

EC24A.A

The value of the alpha component of the pixel.

[VB6]

Byte A

EC24A.C0

The value of the first component of the pixel (red channel).

[VB6]

Byte C0

EC24A.C1

The value of the second component of the pixel (green channel).

[VB6]

Byte C1

EC24A.C2

The value of the third component of the pixel (blue channel).

[VB6]

Byte C2

EC24A.EC24A

Constructs a EC24A object.

[VB6]

```
void EC24A(  
    )  
  
void EC24A(  
    Byte c0,  
    Byte c1,  
    Byte c2  
    )
```

Parameters

c0

The value of the first component of the pixel (red channel).

c1

The value of the second component of the pixel (green channel).

c2

The value of the third component of the pixel (blue channel).

EC24A.Size

The number of bits in a pixel

[VB6]

Size As Long

read-only

EC24A.UINT32Value

-

[VB6]

UINT32Value As Long

read-write

EC24Path Struct

Path from a [EC24](#) image: image pixel coordinates, and associated color pixel value.

PROPERTIES

Pixel	The value of the pixel at this coordinate.
X	Coordinate along the horizontal direction.
Y	Coordinate along the vertical direction.

EC24Path.Pixel

The value of the pixel at this coordinate.

[VB6]

EC24 Pixel

EC24Path.X

Coordinate along the horizontal direction.

[VB6]

Long X

EC24Path.Y

Coordinate along the vertical direction.

[VB6]

Long Y

EC48 Struct

Color pixel value coded as 3 unsigned 16-bit integers (red, green, blue components).

PROPERTIES

BitsPerPixelCode	-
C0	-
C1	-
C2	-
DefaultColorSystem	-
PlanesPerPixel	-
Size	-

METHODS

EC48	-
IsEqual	-

EC48.BitsPerPixelCode

-

[VB6]

BitsPerPixelCode As Integer

read-only

EC48.C0

-

[VB6]

Long C0

EC48.C1

-

[VB6]

Long C1

EC48.C2

-

[VB6]

Long C2

EC48.DefaultColorSystem

-

[VB6]

DefaultColorSystem As EColorSystem

read-only

EC48.EC48

-

[VB6]

```
void EC48(  
    )  
  
void EC48(  
    Long c0,  
    Long c1,  
    Long c2  
    )
```

Parameters

c0

-

c1

-

c2

-

EC48.AreEqual

-

[VB6]

```
Boolean IsEqual(  
    EC48 other  
)
```

Parameters

other

EC48.PlanesPerPixel

-

PlanesPerPixel As Long

read-only

EC48.Size

-

[VB6]

Size As Long

read-only

EColor Struct

Triple of floating-point numbers that encode a color in a given color system.

PROPERTIES

C0	First color component.
C1	Second color component.
C2	Third color component.

METHODS

EColor	Constructor for EColor objects.
--------	---------------------------------

EColor.C0

First color component.

[VB6]
Single C0

EColor.C1

Second color component.

[VB6]

Single C1

EColor.C2

Third color component.

[VB6]

Single C2

EColor.EColor

Constructor for EColor objects.

[VB6]

```
void EColor()
)
void EColor(
    Single c0,
    Single c1,
    Single c2
)
```

Parameters

c0

value for the first color component

c1

value for the second color component

c2

value for the third color component

EDepth16 Struct

Depth value of the pixel, coded as an unsigned 16-bit integer.

PROPERTIES

Size	The number of bits in a pixel
UINT32Value	-
Value	The depth value of the pixel.

METHODS

EDepth16 Constructs a EDepth16 object.

EDepth16.EDepth16

Constructs a EDepth16 object.

```
[VB6]  
void EDepth16()  
void EDepth16(  
    Long value  
)
```

Parameters

value

The depth value of the pixel.

EDepth16.Size

The number of bits in a pixel

[VB6]

Size As Long

read-only

EDepth16.UINT32Value

-

[VB6]

UINT32Value As Long

read-write

EDepth16.Value

The depth value of the pixel.

[VB6]

Long Value

EDepth32f Struct

Depth value of the pixel, coded as an 32-bit float.

PROPERTIES

FLOAT32Value	-
Size	The number of bits in a pixel
Value	The depth value of the pixel.

METHODS

EDepth32f	Constructs a EDepth32f object.
-----------	--------------------------------

EDepth32f.EDepth32f

Constructs a EDepth32f object.

```
[VB6]
void EDepth32f(
)
void EDepth32f(
    Single value
)
```

Parameters

value

The depth value of the pixel.

EDepth32f.FLOAT32Value

-

[VB6]

FLOAT32Value As Single

read-write

EDepth32f.Size

The number of bits in a pixel

[VB6]

Size As Long

read-only

EDepth32f.Value

The depth value of the pixel.

[VB6]

Single Value

EDepth8 Struct

Depth value of the pixel, coded as an unsigned 8-bit integer.

PROPERTIES

<code>Size</code>	The number of bits in a pixel
<code>UINT32Value</code>	-
<code>Value</code>	The depth value of the pixel.

METHODS

<code>EDepth8</code>	Constructs a EDepth8 object.
----------------------	------------------------------

EDepth8.EDepth8

Constructs a EDepth8 object.

```
[VB6]
void EDepth8(
)
void EDepth8(
    Byte value
)
```

Parameters

`value`

The depth value of the pixel.

EDepth8.Size

The number of bits in a pixel

[VB6]

Size As Long

read-only

EDepth8.UINT32Value

-

[VB6]

UINT32Value As Long

read-write

EDepth8.Value

The depth value of the pixel.

[VB6]

Byte Value

EFeatureData Struct

Describes object features.

Remarks

A feature is associated to an array of values, each corresponding to an object of given identification number. A feature is also characterized by the size of the array, a feature number, data size/type information and pointers to both ends of the array. The features can be accessed by their number (see [EFeature](#)). To obtain the value of a given feature of a given object, just use the class member [ECodedImage::GetObjectFeature](#). This enumeration pertains to the EasyObject legacy API and should not be used for new developments. See [ECodedImage2](#) for the new API.

PROPERTIES

FeatDataSize	Data size (see EDataSize).
FeatDataType	Data type (see EDataType).
FeatNum	Code (see EFeature).
Size	Number of objects for which the feature is stored.

EFeatureData.FeatDataSize

Data size (see [EDataSize](#)).

[VB6]

```
EDataSize FeatDataSize
```

EFeatureData.FeatDataType

Data type (see [EDataType](#)).

[VB6]

```
EDataType FeatDataType
```

EFeatureData.FeatNum

Code (see [EFeature](#)).

[VB6]

```
ELegacyFeature FeatNum
```

EFeatureData.Size

Number of objects for which the feature is stored.

[VB6]

```
Long Size
```

EISH Struct

Intensity, Saturation, Hue color system.

PROPERTIES

H	Hue component.
I	Intensity component.
S	Saturation component.

EISH.H

Hue component.

[VB6]

Single H

EISH.I

Intensity component.

[VB6]

Single I

EISH.S

Saturation component.

[VB6]

Single S

ELAB Struct

CIE Lightness, a*, b* color system.

PROPERTIES

A	a* component.
B	b* component.
L	Lightness component.

ELAB.A

a* component.

[VB6]

Single A

ELAB.B

b* component.

[VB6]

Single B

ELAB.L

Lightness component.

[VB6]

Single L

ELCH Struct

Lightness, Chroma, Hue color system.

PROPERTIES

C	Chroma component.
H	Hue component.
L	Lightness component.

ELCH.C

Chroma component.

[VB6]

Single C

ELCH.H

Hue component.

[VB6]

Single H

ELCH.L

Lightness component.

[VB6]

Single L

ELSH Struct

Lightness, Saturation, Hue color system.

PROPERTIES

H	Hue component.
L	Lightness component.

S	Saturation component.
---	-----------------------

ELSH.H

Hue component.

[VB6]

Single H

ELSH.L

Lightness component.

[VB6]

Single L

ELSH.S

Saturation component.

[VB6]

Single S

ELUV Struct

CIE Lightness, u*, v* color system.

PROPERTIES

L	Lightness component.
U	u* component.
V	v* component.

ELUV.L

Lightness component.

[VB6]

Single L

ELUV.U

u* component.

[VB6]

Single U

ELUV.V

v* component.

[VB6]

Single v

EMatchPosition Struct

Represents a single instance of the pattern in the search field, as returned by the EasyMatch matching process.

Remarks

[EMatcher::GetPosition](#) returns instances of this class. A EMatchPosition object represents one matched instance, with all the needed information about it.

PROPERTIES

Angle	Clockwise rotation angle, expressed using the current angle unit, of the pattern found in the image.
AreaRatio	Ratio between the found pattern area inside the search ROI and its complete area.
CenterX	Abscissa of the center of the pattern found in the image.
CenterY	Ordinate of the center of the pattern found in the image.
Interpolated	

Scale	Indicates whether sub-pixel interpolation was actually performed on the position.
Scale factor of the pattern found in the image.	
ScaleX	Measured horizontal scaling of the pattern found in the image, expressed as a dimensionless ratio.
ScaleY	Measured vertical scaling of the pattern found in the image, expressed as a dimensionless ratio.
Score	Indicates how good a matching was.

EMatchPosition.Angle

Clockwise rotation angle, expressed using the current angle unit, of the pattern found in the image.

[VB6]

Single Angle

Remarks

0 if no rotation is allowed.

EMatchPosition.AreaRatio

Ratio between the found pattern area inside the search ROI and its complete area.

[VB6]

Single AreaRatio

EMatchPosition.CenterX

Abscissa of the center of the pattern found in the image.

[VB6]

Single CenterX

EMatchPosition.CenterY

Ordinate of the center of the pattern found in the image.

[VB6]

Single CenterY

EMatchPosition.Interpolated

Indicates whether sub-pixel interpolation was actually performed on the position.

[VB6]

Boolean Interpolated

Remarks

In some cases, when the pattern is found close to the ROI edge, sub-pixel interpolation cannot be used.

EMatchPosition.Scale

Scale factor of the pattern found in the image.

[VB6]

Single Scale

Remarks

1 if no scaling is allowed.

EMatchPosition.ScaleX

Measured horizontal scaling of the pattern found in the image, expressed as a dimensionless ratio.

[VB6]

Single ScaleX

EMatchPosition.ScaleY

Measured vertical scaling of the pattern found in the image, expressed as a dimensionless ratio.

[VB6]

Single ScaleY

EMatchPosition.Score

Indicates how good a matching was.

[VB6]

Single Score

Remarks

- 1** means that the matching was perfect. Lower values correspond to approximate matching; **-1** corresponds to a perfect mismatch (pattern superimposed on its negative image).

EMatrixCodeIso15415GradingParameter s Struct

Holds all grading parameters pertaining to ISO/IEC 15415

PROPERTIES

AxialNonUniformity	-
AxialNonUniformityGrade	-
DecodingGrade	-
FixedPatternDamageGrade	-

GridNonUniformity	-
GridNonUniformityGrade	-
HorizontalPrintGrowth	-
ModulationGrade	-
OverallSymbolGrade	-
ReflectanceMarginGrade	-
SymbolContrast	-
SymbolContrastGrade	-
UnusedErrorCorrection	-
UnusedErrorCorrectionGrade	-
VerticalPrintGrowth	-

EMatrixCode15415GradingParameters.AxialNonUniformity

-

[VB6]

```
Single AxialNonUniformity
```

EMatrixCode15415GradingParameters.AxialNonUniformityGrade

-

[VB6]

```
Long AxialNonUniformityGrade
```

EMatrixCode15415GradingParameters.DecodingGrade

-

[VB6]

```
Long DecodingGrade
```

EMatrixCode15415GradingParameters.FixedPatternDamageGrade

-

[VB6]

```
Long FixedPatternDamageGrade
```

EMatrixCode15415GradingParameters.GridNonUniformity

[VB6]

Single GridNonUniformity

EMatrixCode15415GradingParameters.GridNonUniformityGrade

[VB6]

Long GridNonUniformityGrade

EMatrixCode15415GradingParameters.HorizontalPrintGrowth

[VB6]

Single HorizontalPrintGrowth

EMatrixCode15415GradingParameters.ModulationGrade

[VB6]

```
Long ModulationGrade
```

EMatrixCode15415GradingParameters.OverallSymbolGrade

[VB6]

```
Long OverallSymbolGrade
```

EMatrixCode15415GradingParameters.ReflectanceMarginGrade

[VB6]

```
Long ReflectanceMarginGrade
```

EMatrixCode15415GradingParameters.SymbolContrast

-

[VB6]

Single SymbolContrast

EMatrixCode15415GradingParameters.SymbolContrastGrade

-

[VB6]

Long SymbolContrastGrade

EMatrixCode15415GradingParameters.UnusedErrorCorrection

-

[VB6]

Single UnusedErrorCorrection

EMatrixCodeISO15415GradingParameters.UnusedErrorCorrectionGrade

-

[VB6]

```
Long UnusedErrorCorrectionGrade
```

EMatrixCodeISO15415GradingParameters.VerticalPrintGrowth

-

[VB6]

```
Single VerticalPrintGrowth
```

EMatrixCodeISO29158GradingParameters Struct

Holds all grading parameters pertaining to ISO/IEC 29158

PROPERTIES

CellContrastGrade	-

CellModulationGrade	-
FixedPatternDamageGrade	-
IsMeanLightInRequiredBounds	-
MeanLight	-
MinimumReflectanceGrade	-
OverallSymbolGrade	-

EMatrixCodeIso29158GradingParameters.CellContrastGrade

-

[VB6]

Long CellContrastGrade

EMatrixCodeIso29158GradingParameters.CellModulationGrade

-

[VB6]

```
Long CellModulationGrade
```

EMatrixCode1so29158GradingParameters.FixedPatternDamageGrade

[VB6]

```
Long FixedPatternDamageGrade
```

EMatrixCode1so29158GradingParameters.IsMeanLightInRequiredBounds

[VB6]

```
Boolean IsMeanLightInRequiredBounds
```

EMatrixCode1so29158GradingParameters.MeanLight

[VB6]

Single MeanLight

EMatrixCodeIso29158GradingParameters.Minimum
ReflectanceGrade

-

[VB6]

Long MinimumReflectanceGrade

EMatrixCodeIso29158GradingParameters.OverallSy
mbolGrade

-

[VB6]

Long OverallSymbolGrade

EMatrixCodeSemiT10GradingParameters
Struct

Holds all grading parameters pertaining to Semi T10-

PROPERTIES

CellDefects	-
DataMatrixCellHeight	-
DataMatrixCellWidth	-
FinderPatternDefects	-
HorizontalMarkGrowth	-
HorizontalMarkMisplacement	-
SymbolContrast	-
SymbolContrastSNR	-
UnusedErrorCorrection	-
VerticalMarkGrowth	-
VerticalMarkMisplacement	-

EMatrixCodeSemiT10GradingParameters.CellDefect
S

-

[VB6]

```
Single CellDefects
```

EMatrixCodeSemiT10GradingParameters.DataMatrixCellHeight

[VB6]

```
Single DataMatrixCellHeight
```

EMatrixCodeSemiT10GradingParameters.DataMatrixCellWidth

[VB6]

```
Single DataMatrixCellWidth
```

EMatrixCodeSemiT10GradingParameters.FinderPatternDefects

[VB6]

Single FinderPatternDefects

EMatrixCodeSemiT10GradingParameters.HorizontalMarkGrowth

-

[VB6]

Single HorizontalMarkGrowth

EMatrixCodeSemiT10GradingParameters.HorizontalMarkMisplacement

-

[VB6]

Single HorizontalMarkMisplacement

EMatrixCodeSemiT10GradingParameters.SymbolContrast

-

[VB6]

Single SymbolContrast

EMatrixCodeSemiT10GradingParameters.SymbolContrastSNR

-

[VB6]

Single SymbolContrastSNR

EMatrixCodeSemiT10GradingParameters.UnusedErrorCorrection

-

[VB6]

Single UnusedErrorCorrection

EMatrixCodeSemiT10GradingParameters.VerticalMarkGrowth

-

[VB6]

Single VerticalMarkGrowth

EMatrixCodeSemiT10GradingParameters.VerticalMarkMisplacement

-

[VB6]

Single VerticalMarkMisplacement

EObjectData Struct

Describes objects.

Remarks

An object is characterized by a class, a unique identification number, the number of its constituent runs, the number of its holes (if the object is a real object, not a hole), a selection flag, an identification flag (real object or hole) and the list of its constituent runs. After the object construction phase (real objects and eventually holes), all the objects are gathered in a single dynamic list. The objects can be accessed by their absolute position in the list as well as by their identification number. This structure pertains to the EasyObject legacy API and should not be used for new developments.

Note. After a sorting operation, the objects retain their identification number, not their absolute position in the list. If need be, the list of runs of an object can be traversed by means of the following functions: [GetObjNbRun](#), [ECodedImage::GetObjFirstRunPtr](#), [ECodedImage::GetObjLastRunPtr](#).

PROPERTIES

<code>Class</code>	Class code.
<code>IsHole</code>	TRUE if the object is a hole, FALSE otherwise.
<code>IsSelected</code>	Selection flag.
<code>ObjNbHole</code>	Number of holes.
<code>ObjNbRun</code>	Number of runs.
<code>ObjNum</code>	Identification number.

EObjectData.Class

Class code.

[VB6]
Long Class

EObjectData.IsHole

TRUE if the object is a hole, **FALSE** otherwise.

[VB6]
Boolean IsHole

EObjectData.isSelected

Selection flag.

[VB6]

```
Byte IsSelected
```

EObjectData.ObjNbHole

Number of holes.

[VB6]

```
Long ObjNbHole
```

EObjectData.ObjNbRun

Number of runs.

[VB6]

```
Long ObjNbRun
```

EObjectData.ObjNum

Identification number.

[VB6]

Long ObjNum

EOCR2CharacterCandidate Struct

Holds a single recognition score for a detected character from the image

Remarks

The variable "code" contains the ASCII-representation of the reference character from the database. The variable "score" contains the recognition score between the detected character and the reference character.

PROPERTIES

Code	Contains the ASCII-representation of the reference character from the database.
Score	Contains the recognition score between the detected character and the reference character.

METHODS

EOCR2CharacterCandidate	Constructs an EOCR2CharacterCandidate context.
--------------------------------	--

EOCR2CharacterCandidate.Code

Contains the ASCII-representation of the reference character from the database.

[VB6]

Long Code

EOCR2CharacterCandidate.EOCR2CharacterCandidate

Constructs an EOCR2CharacterCandidate context.

```
[VB6]  
void EOCR2CharacterCandidate()  
)  
  
void EOCR2CharacterCandidate(  
Long code,  
Single score  
)
```

Parameters

code

-

score

-

EOCR2CharacterCandidate.Score

Contains the recognition score between the detected character and the reference character.

```
[VB6]
```

Single Score

EPath Struct

Path from an image: image pixel coordinates.

PROPERTIES

X	Coordinate along the horizontal direction.
Y	Coordinate along the vertical direction.

EPath.X

Coordinate along the horizontal direction.

[VB6]

Long X

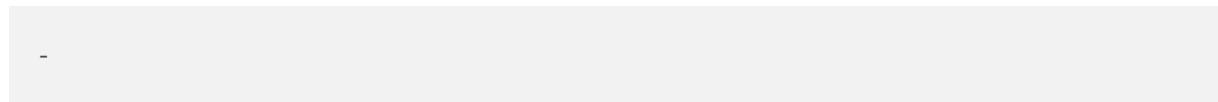
EPath.Y

Coordinate along the vertical direction.

[VB6]

Long Y

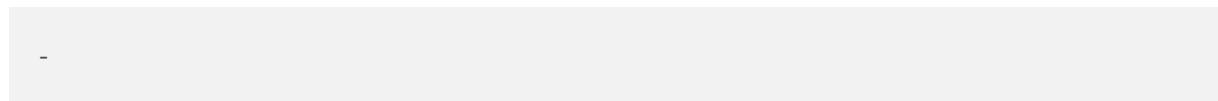
EPeak Struct



PROPERTIES

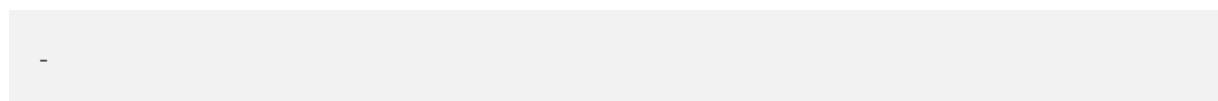
Amplitude	-
Area	-
Center	-
Length	-
Start	-

EPeak.Amplitude



[VB6]
Long Amplitude

EPeak.Area



[VB6]

Long Area

EPeak.Center

-

[VB6]

Single Center

EPeak.Length

-

[VB6]

Long Length

EPeak.Start

-

[VB6]

Long Start

ERGB Struct

NTSC/PAL/SMPTE Red, Green, Blue color system.

PROPERTIES

B	Blue component.
G	Green component.
R	Red component.

ERGB.B

Blue component.

[VB6]
Single B

ERGB.G

Green component.

[VB6]
Single G

ERGB.R

Red component.

[VB6]

Single R

ERGBColor Struct

NTSC/PAL/SMPTE Red, Green, Blue color system.

PROPERTIES

Blue	Blue component.
Green	Green component.
Red	Red component.

METHODS

ERGBColor	Constructs an ERGBColor object.
-----------	---------------------------------

ERGBColor.Blue

Blue component.

[VB6]

Long Blue

ERGBColor.ERGBColor

Constructs an ERGBColor object.

[VB6]

```
void ERGBColor(
    Long red,
    Long green,
    Long blue
)
void ERGBColor(
)
```

Parameters

red

Red component.

green

Green component.

blue

Blue component.

ERGBColor.Green

Green component.

[VB6]

Long Green

ERGBColor.Red

<p>Red component.</p>

<p>[VB6]</p>

<p>Long Red</p>

ERunData Struct

<p>Describes runs.</p>

Remarks

A run is characterized by a starting point (**OrgX**, **OrgY**), by a length, a class, a unique identification number and the number of the object to which they belong. After the run construction phase, all the runs are gathered in a single dynamic list.

PROPERTIES

Class	Class.
Len	Length.
ObjNum	Identification number of the object to which the run belongs.
OrgX	Start point abscissa.
OrgY	Start point ordinate.

ERunData.Class

Class.

[VB6]

Long Class

ERunData.Len

Length.

[VB6]

Long Len

ERunData.ObjNum

Identification number of the object to which the run belongs.

[VB6]

Long ObjNum

ERunData.OrgX

Start point abscissa.

[VB6]

Long OrgX

ERunData.OrgY

Start point ordinate.

[VB6]

Long OrgY

ETransitionData Struct

The transition data is a structure containing information about a transition. To recuperate a transition data, use the GetTransitionData(UINT32 index = ~0) method. If the parameter of the method is equal to ~0, the transition data designed by the Picking Index will be returned following the [EPickingMode](#) if not equal to [EPickingMode_All](#); in the case of [EPickingMode_All](#), the first transition will be returned.

PROPERTIES

Contrast	Difference between the gray level in one side of the transition and the gray level in the other side.
Location	Position of the transition along the axes of the gauge relative to its center in pixels.
MaxSlope	Maximum slope in gray levels/pixel.

Polarity	Black to white (+1) or white to black (-1) in the direction of the point gauge.
Score	The score is computed after detection. The score is intended to help the user in sorting "good" and "bad" transitions.
Width	Number of pixels needed to go from one side of the transition to the other side.

ETransitionData.Congtrast

Difference between the gray level in one side of the transition and the gray level in the other side.

[VB6]

Single Contrast

ETransitionData.Location

Position of the transition along the axes of the gauge relative to its center in pixels.

[VB6]

Single Location

ETransitionData.MaxSlope

Maximum slope in gray levels/pixel.

[VB6]

Single MaxSlope

ETransitionData.Polarity

Black to white (+1) or white to black (-1) in the direction of the point gauge.

[VB6]

Long Polarity

ETransitionData.Score

The score is computed after detection. The score is intended to help the user in sorting "good" and "bad" transitions.

[VB6]

Single Score

ETransitionData.Width

Number of pixels needed to go from one side of the transition to the other side.

[VB6]

Long Width

EVSH Struct

Value, Saturation, Hue color system.

PROPERTIES

H	Hue component.
S	Saturation component.
V	Value component.

EVSH.H

Hue component.

[VB6]

Single H

EVSH.S

Saturation component.

[VB6]

Single S

EVSH.V

Value component.

[VB6]

Single V

EXYZ Struct

CIE XYZ color system.

PROPERTIES

X	X component.
Y	Y component.
Z	Z component.

EXYZ.X

X component.

[VB6]

Single X

EXYZ.Y

Y component.

[VB6]

Single Y

EXYZ.Z

Z component.

[VB6]

Single Z

EYIQ Struct

CCIR Luma, Inphase, Quadrature color system.

PROPERTIES

I	Inphase component.
Q	Quadrature component.

Y	Luma component.
---	-----------------

EYIQ.I

Inphase component.

[VB6]	Single I
-------	----------

EYIQ.Q

Quadrature component.

[VB6]	Single Q
-------	----------

EYIQ.Y

Luma component.

[VB6]	Single Y
-------	----------

EYSH Struct

CCIR Luma, Saturation, Hue color system.

PROPERTIES

H	Hue component.
S	Saturation component.
Y	Luma component.

EYSH.H

Hue component.

[VB6]

Single H

EYSH.S

Saturation component.

[VB6]

Single S

EYSH.Y

Luma component.

[VB6]

Single Y

EYUV Struct

CCIR Luma, U Chroma, V Chroma color system.

PROPERTIES

U	U Chroma component.
V	V Chroma component.
Y	Luma component.

EYUV.U

U Chroma component.

[VB6]

Single U

EYUV.V

V Chroma component.

[VB6]

Single V

EYUV.Y

Luma component.

[VB6]

Single Y

Enumerations

EAdaptiveThresholdMethod Enum

Adaptive thresholding modes.

EAdaptiveThresholdMethod_Mean	Use the mean as threshold.
EAdaptiveThresholdMethod_Median	Use the median as threshold.
EAdaptiveThresholdMethod_Middle	Use the middle of the values interval as threshold.

EAngleUnit Enum

The angle units that are supported by Open eVision.

EAngleUnit_Revolutions	Revolutions (0..1 corresponds to a full revolution).
EAngleUnit_Radians	Radians (0..2Pi corresponds to a full revolution).
EAngleUnit_Degrees	Degrees (0..360 corresponds to a full revolution).
EAngleUnit_Grades	Grades (0..400 corresponds to a full revolution).

EAithmeticLogicOperation Enum

Supported arithmetic or logic pixel-wise operators.

EAithmeticLogicOperation_Copy	Sheer copy.
EAithmeticLogicOperation_Invert	Complement. (*)
EAithmeticLogicOperation_Add	Saturated addition. (*)
EAithmeticLogicOperation_Subtract	Saturated subtraction. (*)
EAithmeticLogicOperation_Multiply	Saturated multiplication. (*)
EAithmeticLogicOperation_Divide	Saturated division. (*)
EAithmeticLogicOperation_Modulo	Modulo. (*)
EAithmeticLogicOperation_ShiftLeft	Arithmetic left shift (multiplication by a power of 2). (*)
EAithmeticLogicOperation_ShiftRight	Arithmetic right shift (division by a power of 2). (*)
EAithmeticLogicOperation_ScaledAdd	Non saturating addition $((left + right) / 2)$. (*)
EAithmeticLogicOperation_ScaledSubtract	Non saturating subtraction $((left + complemented right) / 2)$. (*)

EArithmeticLogicOperation_ScaledMultiply	Non saturating multiplication (left * right / 256 in the BW8 case, and left * right / 65,536 in the BW16 one). (*)
EArithmeticLogicOperation_ScaledDivide	Non saturating division (256 * left / right in the BW8 case, and 65,536 * left / right in the BW16 one). (*)
EArithmeticLogicOperation_BitwiseAnd	Bitwise AND.
EArithmeticLogicOperation_BitwiseOr	Bitwise OR.
EArithmeticLogicOperation_BitwiseXor	Bitwise exclusive OR.
EArithmeticLogicOperation_LogicalAnd	Logical AND (non zero is TRUE). (*)
EArithmeticLogicOperation_LogicalOr	Logical OR (non zero is TRUE). (*)
EArithmeticLogicOperation_LogicalXor	Logical exclusive OR (non zero is TRUE). (*)
EArithmeticLogicOperation_Min	Minimum. (*)
EArithmeticLogicOperation_Max	Maximum. (*)
EArithmeticLogicOperation_SetZero	Copy the right operand where the left operand is zero.
EArithmeticLogicOperation_SetNonZero	Copy the right operand where the left operand is non zero.
EArithmeticLogicOperation_Equal	Equality comparison. (*)

EArithmeticLogicOperation_NotEqual	Non equality comparison. (*)
EArithmeticLogicOperation_GreaterOrEqual	"Greater or equal" comparison. (*)
EArithmeticLogicOperation_LesserOrEqual	"Lesser or equal" comparison.
EArithmeticLogicOperation_Greater	"Greater" comparison. (*)
EArithmeticLogicOperation_Lesser	"Lesser" comparison. (*)
EArithmeticLogicOperation_Compare	Absolute value of the difference. (*)
EArithmeticLogicOperation_Overlay	Overlay of one image onto a source image giving a destination image. (See note at the end of the topic). (*) (**)
EArithmeticLogicOperation_BitwiseNot	Same as EArithmeticLogicOperation_Invert .
EArithmeticLogicOperation_Average	Same as EArithmeticLogicOperation_ScaledAdd . (*)

Remarks

(*) Not applicable for the **BW1** images/ROIs. (**) In the overlay image, black pixels (0 valued) are considered as transparent. If a **C24** image is used as overlay, all pixels (but the black ones) will be copied to the destination image. If a **BW8** image is used as overlay, all non-black pixels will be converted to the color defined by the **OverlayColor** parameter before copy to the destination image. The destination image is always a **C24** image. If no source image is given (only overlay and destination), the destination image is considered as the source image.

EasyOCR2CharacterFilter Enum

This enumeration contains the possible filters for loading fonts in easyOCR2.

EasyOCR2CharacterFilter_ASCII	All ASCII characters are loaded.
EasyOCR2CharacterFilter_Letters	Only (alphabetic) letters are loaded.
EasyOCR2CharacterFilter_Digits	Only digits are loaded.

EasyOCR2CharSpacingBias Enum

This enumeration contains the possible biases for the optimised character spacing in the detection phase of easyOCR2.

EasyOCR2CharSpacingBias_Wide	The optimisation is biased toward wide character spacing.
EasyOCR2CharSpacingBias_Neutral	The optimisation is not biased.
EasyOCR2CharSpacingBias_Narrow	The optimisation is biased toward narrow character spacing.

EasyOCR2CharWidthBias Enum

This enumeration contains the possible biases for the optimised character width in the detection phase of easyOCR2.

EasyOCR2CharWidthBias_Widest	The optimisation is biased toward very wide boxes.
EasyOCR2CharWidthBias_Wide	The optimisation is biased toward wide boxes.
EasyOCR2CharWidthBias_Neutral	The optimisation is not biased.
EasyOCR2CharWidthBias_Narrow	The optimisation is biased toward narrow boxes.
EasyOCR2CharWidthBias_Narrowest	The optimisation is biased toward very narrow boxes.

EasyOCR2DrawDetectionStyle Enum

This enumeration contains the possible drawing styles for the detection results in easyOCR2.

EasyOCR2DrawDetectionStyle_DrawChars	A bounding box is drawn around each individual detected character
EasyOCR2DrawDetectionStyle_DrawWords	A bounding box is drawn around each detected word, containing all characters in the word
EasyOCR2DrawDetectionStyle_DrawLines	A bounding box is drawn around each detected line, containing all characters/words in the line

EasyOCR2DrawDetectionStyle_

DrawText

EasyOCR2DrawRecognitionStyle Enum

A bounding box is drawn around the detected text, containing all characters/words/lines in the text

t
ionStyle Enum

This enumeration contains the possible drawing styles for the recognition results in easyOCR2.

EasyOCR2DrawRecognitionStyle_LeftTop	The recognition result is drawn at the left top of the character
EasyOCR2DrawRecognitionStyle_LeftMiddle	The recognition result is drawn at the left middle of the character
EasyOCR2DrawRecognitionStyle_LeftBottom	The recognition result is drawn at the left bottom of the character
EasyOCR2DrawRecognitionStyle_BottomLeft	The recognition result is drawn at the bottom left of the character
EasyOCR2DrawRecognitionStyle_BottomMiddle	The recognition result is drawn at the bottom middle of the character
EasyOCR2DrawRecognitionStyle_BottomRight	The recognition result is drawn at the bottom right of the character

EasyOCR2DrawRecognitionStyle_RightBottom	The recognition result is drawn at the right bottom of the character
EasyOCR2DrawRecognitionStyle_RightMiddle	The recognition result is drawn at the right middle of the character
EasyOCR2DrawRecognitionStyle_RightTop	The recognition result is drawn at the right top of the character
EasyOCR2DrawRecognitionStyle_TopRight	The recognition result is drawn at the top right of the character
EasyOCR2DrawRecognitionStyle_TopMiddle	The recognition result is drawn at the top middle of the character
EasyOCR2DrawRecognitionStyle_TopLeft	The recognition result is drawn at the top left of the character

EasyOCR2DrawSegmentationStyle Enum

This enumeration contains the possible drawing styles for the segmentation results in easyOCR2.

EasyOCR2DrawSegmentationStyle_DrawBlobs	The segmented blobs are drawn directly.
---	---

EasyOCR2TextPolarity Enum

This enumeration contains the possible polarities of text searched during segmentation.

EasyOCR2TextPolarity_WhiteOnBlack	The text is light on a dark background
EasyOCR2TextPolarity_BlackOnWhite	The text is dark on a light background

ECalibrationMode Enum

Allowed values for the calibration mode.

ECalibrationMode_Raw	No calibration at all.
ECalibrationMode_Inverse	The ordinate axis points downwards instead of upwards.
ECalibrationMode_Scaled	The pixels are assigned a physical size.
ECalibrationMode_Anisotropic	The physical size of the pixels differ horizontally and vertically. (Beware there is a restriction pertaining to the allowed image anisotropy. The resulting pixels aspect ratio should be in the range [-4/3, -3/4] (or [3/4, 4/3]), otherwise the calibration process could fail).
ECalibrationMode_Skewed	

	The coordinate axis make an angle with the image edge.
ECalibrationMode_Tilted	The field-of-view plane is not perpendicular to the optical axis.
ECalibrationMode_Radial	The lens introduces some amount of radial distortion.
ECalibrationMode_Bilinear	This mode can not be combined with other calibration mode. The bilinear calibration is based on a first order polynomial approach.
ECalibrationMode_Quadratic	This mode can not be combined with other calibration mode. The quadratic calibration is based on a second order polynomial approach.

ECannyThresholdingMode Enum

The thresholding modes for the Canny edge detector.

ECannyThresholdingMode_Relative	Relative thresholding mode.
ECannyThresholdingMode_Absolute	Absolute thresholding mode.

ECharCreationMode Enum

Allowed values for the character creation mode in EasyOCV.

ECharCreationMode_Group	Form a single character comprising all blobs.
ECharCreationMode_Overlap	Form single characters out of blobs whose bounding box overlap.
ECharCreationMode_Separate	Form one character for each blob.

EClippingMode Enum

Allows to choose how the fitted segment length and centre are computed

EClippingMode_CenteredNominal	Regular mode: the fitted segment always has the nominal length of the line gauge.
EClippingMode_ClippedToValidSamples	The fitted segment does not extend beyond valid samples. It is clipped to the projection of the valid samples on the fitted line.
EClippingMode_ClippedInNominalShape	The segment is built by clipping the fitted line in the rectangular range where the ELineGauge looks for valid transition samples, i.e. the rectangle which is centered and aligned on the ELineGauge nominal line, and which height is two times the ELineGauge::Tolerance .

EColorQuantization Enum

Allowed values for the quantization mode in EasyColor.

EColorQuantization_FullRange	Values are quantized in range 0..255 .
------------------------------	---

EColorQuantization_Ccir601	Values are quantized in range 16..235 for the R, G, B or Y component, and in range 16..240 for the I, Q, U and y components.
----------------------------	--

Remarks

When quantizing the color values for the RGB or YIQ/YUV representation, one usually uses the full **0..255** range. Anyway, the CCIR has defined an alternate convention such that some values in this interval are reserved. Before performing a conversion, functions [EasyColor::SrcQuantization](#) and [EasyColor::DstQuantization](#) can be used to specify the rule used.

EColorSystem Enum

The color systems that are supported by Open eVision.

EColorSystem_NoColor	Undefined
EColorSystem_Bilevel	Binary black & white.
EColorSystem_GrayLevel	Continuous tone black & white.
EColorSystem_Xyz	CIE XYZ.
EColorSystem_Rgb	NTSC/PAL/SMPTE Red, Green, Blue.
EColorSystem_Lab	CIE Lightness, a*, b*.
EColorSystem_Luv	CIE Lightness, u*, v*.
EColorSystem_Yuv	CCIR Luma, U Chroma, V Chroma.
EColorSystem_Yiq	CCIR Luma, Inphase, Quadrature.

EColorSystem_Lch	Lightness, Chroma, Hue.
EColorSystem_Ish	Intensity, Saturation, Hue
EColorSystem_Lsh	Lightness, Saturation, Hue.
EColorSystem_Vsh	Value, Saturation, Hue.
EColorSystem_Ysh	CCIR Luma, Saturation, Hue.

Remarks

Open eVision supports several color systems. The achromatic ones are related to black and white and gray-level images ([EImageBW1](#) and [EImageBW8](#)). The remaining ones apply to color images ([EImageC24](#)). Also see unquantized and quantized colors for the allowed ranges of values.

EConnexity Enum

Possible values for the connexity of a contour.

EConnexity_Connexity4	Pixels touching by an edge are considered connected.
EConnexity_Connexity8	Pixels touching by an edge or a corner are considered connected.

EContourMode Enum

Possible modes for contour traversal.

EContourMode_Clockwise	The contour is traversed clockwise.
EContourMode_ClockwiseAlwaysClosed	The contour is traversed clockwise and the image border may be followed if necessary to close the contour.
EContourMode_ClockwiseContinuelfBorder	Contour traversal is restarted counterclockwise when an image border is met.
EContourMode_Anticlockwise	The contour is traversed counterclockwise.
EContourMode_AnticlockwiseContinuelfBorder	Contour traversal is restarted clockwise when an image border is met.
EContourMode_AnticlockwiseAlwaysClosed	The contour is traversed anticlockwise and the image border may be followed if necessary to close the contour.

EContourThreshold Enum

Allowed thresholding modes for contour traversal.

EContourThreshold_Above	Traverse the pixels just above the threshold.
EContourThreshold_Below	Traverse the pixels just below the threshold.

ECorrelationMode Enum

Allowed values for the EasyMatch correlation mode.

<code>ECorrelationMode_Standard</code>	Correlation sensitive to changes in intensity and/or contrast (computed from the raw image/pattern gray values).
<code>ECorrelationMode_OffsetNormalized</code>	Correlation made unsensitive to changes in intensity (computed from the centered image/pattern gray values).
<code>ECorrelationMode_GainNormalized</code>	Correlation made unsensitive to changes in contrast (computed from the reduced image/pattern gray values).
<code>ECorrelationMode_Normalized</code>	Correlation made unsensitive to changes in both intensity and contrast (computed from the centered and reduced image/pattern gray values). Default mode.

EDataSize Enum

Possible data sizes for an object feature.

<code>EDataSize_BitsPerPixel1</code>	bit
<code>EDataSize_BitsPerPixel8</code>	byte

EDataSize_BitsPerPixel16	word
EDataSize_BitsPerPixel32	long word
EDataSize_BitsPerPixel64	quad word
EDataSize_BitsPerPixel24	3 bytes
EDataSize_BitsPerPixel96	12 bytes

EDataType Enum

Possible data types for an object feature.

EDataType_UnsignedInt	Unsigned integer.
EDataType_SignedInt	Signed integer.
EDataType_Float	Floating-point.

EDegreesOfFreedom Enum

Allowed values for the degrees of freedom in [EChecker](#).

EDegreesOfFreedom_Translation	Translation allowed.
EDegreesOfFreedom_Rotation	Rotation allowed.

EDegreesOfFreedom_Scaling

Scaling allowed.

E

D

Diagnostic Enum

Possible defect codes in EasyOCV.

EDiagnostic_CharNotFound	Insufficient location score for a character.
EDiagnostic_CharOverprinting	Too much inking on a character.
EDiagnostic_CharUnderprinting	Too little inking on a character.
EDiagnostic_CharMismatch	Wrong character shape.
EDiagnostic_TextNotFound	Insufficient location score for a text.
EDiagnostic_TextOverprinting	Too much inking on a text.
EDiagnostic_TextUnderprinting	Too little inking on a text.
EDiagnostic_TextMismatch	Wrong text shape.
EDiagnostic_BadContrast	Global contrast (of inspection ROI) out of range.
EDiagnostic_Undefined	-

EDoubleThresholdMode Enum

The double threshold mode for the selection of coded elements with respect to a given feature.

EDoubleThresholdMode_Inside	The value of the feature must be greater or equal to the low threshold, and strictly less than the high threshold.
EDoubleThresholdMode_Outside	The value of the feature must be strictly less than the low threshold, or greater or equal to the high threshold.

EDraggingMode Enum

Defines how the shape could be dragged

EDraggingMode_Standard	Allows positioning the shape edges symmetrically.
EDraggingMode_ToEdges	Allows positioning each shape edge individually.

EDragHandle Enum

Allowed values for a handle identifier in the context of handle dragging.

EDragHandle_NoHandle	No handle.
EDragHandle_Inside	Inside handle.
EDragHandle_North	Northern handle.
EDragHandle_East	Eastern handle.
EDragHandle_South	Southern handle.
EDragHandle_West	Western handle.
EDragHandle_NorthWest	North-Western handle.
EDragHandle_SouthWest	South-Western handle.
EDragHandle_NorthEast	North-Eastern handle.
EDragHandle_SouthEast	South-Eastern handle.
EDragHandle_Center	Circle, rectangle or wedge center handle.
EDragHandle_Org	Line segment or circle arc origin handle.
EDragHandle_Mid	Line segment or circle arc middle handle.
EDragHandle_End	Line segment or circle arc end handle.
EDragHandle_Tol0	First tolerance handle.
EDragHandle_Tol1	Second tolerance handle.
EDragHandle_Tol_x0	Rectangle leftmost first tolerance handle.

EDragHandle_Tol_x1	Rectangle leftmost second tolerance handle.
EDragHandle_Tol_y0	Rectangle lower first tolerance handle.
EDragHandle_Tol_y1	Rectangle lower second tolerance handle.
EDragHandle_Tol_XX0	Rectangle rightmost first tolerance handle.
EDragHandle_Tol_XX1	Rectangle rightmost second tolerance handle.
EDragHandle_Tol YY0	Rectangle upper first tolerance handle.
EDragHandle_Tol YY1	Rectangle upper second tolerance handle.
EDragHandle_Tol_a0	Wedge leftmost first tolerance handle.
EDragHandle_Tol_a1	Wedge leftmost second tolerance handle.
EDragHandle_Tol_AA0	Wedge rightmost first tolerance handle.
EDragHandle_Tol_AA1	Wedge rightmost second tolerance handle.
EDragHandle_Tol_r0	Wedge inner first tolerance handle.
EDragHandle_Tol_r1	Wedge inner second tolerance handle.
EDragHandle_Tol_RR0	Wedge outer first tolerance handle.
EDragHandle_Tol_RR1	Wedge outer second tolerance handle.
EDragHandle_Edge_x	Rectangle leftmost edge handle.
EDragHandle_Edge_XX	Rectangle rightmost edge handle.

EDragHandle_Edge_y	Rectangle lower edge handle.
EDragHandle_Edge_YY	Rectangle upper edge handle.
EDragHandle_Edge_a	Wedge leftmost edge handle.
EDragHandle_Edge_AA	Wedge rightmost edge handle.
EDragHandle_Edge_r	Wedge outer edge handle.
EDragHandle_Edge_RR	Wedge inner edge handle.

EDrawableFeature Enum

The various features that can be drawn for coded elements.

EDrawableFeature_BoundingBox	The bounding box.
EDrawableFeature_ConvexHull	The convex hull.
EDrawableFeature_Ellipse	The ellipse of inertia.
EDrawableFeature_FeretBox22	The Feret box oriented at 22.5°.
EDrawableFeature_FeretBox45	The Feret box oriented at 45°.
EDrawableFeature_FeretBox68	The Feret box oriented at 67.5°.
EDrawableFeature_GravityCenter	The gravity center.
EDrawableFeature_	

MinimumEnclosingRectangle	The minimum enclosing rectangle.
EDrawableFeature_FeretBox	The Feret box oriented at a fixed angle. Only available for selections of coded elements: The angle of interest is set through the EObjectSelection::FeretAngle property.
EDrawableFeature_WeightedGravityCenter	The gravity center of the pixels of the attached image over the coded element. Only available for selections of coded elements: The attached image is set through the EObjectSelection::AttachedImage property.

EDrawingMode Enum

Allowed modes to draw the bounding box of a symbol.

EDrawingMode_Nominal	Draws the nominal point location or model fitting gauge.
EDrawingMode_Actual	Draws the located point or the fitted model.
EDrawingMode_SampledPaths	Draws the sampled segments along the model.
EDrawingMode_SampledPath	Draws the sampled segment specified by ELineGauge::MeasureSample .
EDrawingMode_PointsInSkipRange	Draws the skipped sampled points in addition to the non-skipped points.
EDrawingMode_SampledPoints	Draws the sampled points along the model.
EDrawingMode_SampledPoint	Draws the sampled point specified by ELineGauge::MeasureSample .

EDrawingMode_InvalidSampledPoints	Draws the invalid sampled points along the model.
EDrawingMode_Learn	Draw the pattern learning ROI(s).
EDrawingMode_Match	Draw the pattern searching ROI(s).
EDrawingMode_Position	Draw the found pattern ROI(s).
EDrawingMode_Inspected	Draw the inspected ROI.
EDrawingMode_MaxInspected	Draw the largest possible inspected ROI.

EEncodingConnexity Enum

The connexity mode for the encoding process.

EEncodingConnexity_Four	Pixels touching by an edge are considered connected.
EEncodingConnexity_Eight	Pixels touching by an edge or a corner are considered connected.

EError Enum

Possible Open eVision error codes.

EError_Ok	Success

EError_EndOfImageSequence	End of image sequence
EError_UserDialogFailed	User dialog failed
EError_ImageLimitsReached	Image limits reached
EError_InvalidAsciiPadding	Invalid ASCII padding
EError_InvalidOperation	Invalid operation - See Reference
EError_InvalidBitsPerPixel	Invalid depth (bits per pixel) - Check type compatibility
EError_InvalidDataType	Invalid data type - Check type compatibility
EError_InvalidDataSize	Invalid data size - Check type compatibility
EError_ParametersOutOfRange	Parameters out of range - See Reference
EError_InvalidMode	Invalid mode - See Reference
EError_EndSmallerThanStart	End smaller than start - Adjust indices
EError_Parameter1OutOfRange	Parameter 1 out of range - See Reference
EError_Parameter2OutOfRange	Parameter 2 out of range - See Reference
EError_Parameter3OutOfRange	Parameter 3 out of range - See Reference
EError_Parameter4OutOfRange	Parameter 4 out of range - See Reference
EError_Parameter5OutOfRange	Parameter 5 out of range - See Reference
EError_Parameter6OutOfRange	Parameter 6 out of range - See Reference

EError_Parameter7OutOfRange	Parameter 7 out of range - See Reference
EError_Parameter8OutOfRange	Parameter 8 out of range - See Reference
EError_WindowsError	Out of GDI handles
EError_InvalidPlanesPerPixel	Invalid planes per pixel - Check image type or file contents
EError_BW1ImageExpected	1 bit black & white (BW1) image expected - Check image type or file contents
EError_BW8ImageExpected	8 bits black & white (BW8) image expected - Check image type or file contents
EError_BW16ImageExpected	16 bits black & white (BW16) image expected - Check image type or file contents
EError_BW32ImageExpected	32 bits black & white (BW32) image expected - Check image type or file contents
EError_TemplateCallNeedsSpecialization	Template call needs specialization
EError_CannotCreateMutex	Cannot create Mutex
EError_CannotLockMutex	Cannot lock Mutex
EError_CannotUnlockMutex	Cannot unlock Mutex
EError_CannotDeleteMutex	Cannot delete Mutex
EError_TimeoutReached	Timeout reached
EError_FunctionNotFound	Function not found

EError_CopyNotAllowed	Copy not allowed
EError_SingularMatrix	Internal error (code: 1050)
EError_DivisionByZero	Division by zero - Check parameters
EError_ReadOnlyProperty	This property is read-only and cannot be accessed
EError_UndefinedProperty	This property is undefined and cannot be accessed
EError_ItemNotFound	The data structure does not contain the specified item
EError_NextItemNotFound	The specified item has no next sibling
EError_FileAccessProblems	File access problems - Check file pathname and device state
EError_FileCouldNotBeOpened	File could not be opened - Check file pathname, troubleshoot disk device
EError_FilwhileReading	File error while reading - Check file integrity, troubleshoot disk device
EError_FilwhileWriting	File error while writing - Check free disk space, troubleshoot disk device
EError_BadFileFormat	Bad file format - Check file source or contents
EError_FileCouldNotBeClosed	File could not be closed - Check free disk space, troubleshoot disk device
EError_UnsupportedFileFormatVersion	Unsupported file format version - Upgrade to newer release
EError_	

MissingOrUnsupportedFileExtension	Missing or unsupported file extension - Check file name/format match
EError_FileIsReadOnly	File is read-only - Set to read-write or save under another name
EError_UnsupportedObjectTypeInArchive	The archive does not contain the correct object type - Check your archiving routines
EError_UnknownArchiveError	An unexpected error occurred during archive access
EError_SerializerShouldBeInReadMode	Attempting to read with a serializer not open for read access
EError_SerializerShouldBeInWriteMode	Attempting to write with a serializer not open for write access
EError_FileExists	Attempting to overwrite an existing file while not allowed to do so
EError_SerializerNotOpen	Attempting to use a serializer that is not correctly opened
EError_UnrecognizedFileFormat	Unrecognized File Format
EError_WrongColorFormatFileFormatCombination	Wrong Color Format File Format Combination
EError_FileDoesNotExist	Attempting to open a file which doesn't exist
EError_ObjectTooLargeToBeSerialized	Object is too large to be serialized
EError_UnsupportedFileFormat	Unsupported File Format

EError_UnsupportedTiffFormat	Unsupported TIFF format - Convert TIFF file
EError_UnsupportedBmpFormat	Unsupported BMP format - Convert BMP file
EError_UnsupportedJpegFormat	Unsupported JPEG format - Convert JPEG file
EError_BilevelImageExpected	Bi-level image expected - Use BW1
EError_GrayLevelImageExpected	Gray-level image expected - Use BW8
EError_ColorImageExpected	Color image expected - Use C24
EError_BilevelFormatExpected	Bi-level format expected - Convert file to black & white
EError_GrayLevelFormatExpected	Gray-level format expected - Convert file to gray shades
EError_ColorFormatExpected	Color format expected - Convert file to true color
EError_CannotReadJpegFile	Cannot read JPEG file - Troubleshoot disk device
EError_CannotWriteJpegFile	Cannot write JPEG file - Troubleshoot disk device
EError_WrongFileExtension	Wrong file extension
EError_UnableToAllocateTemporaryMemory	Unable to allocate temporary memory - Fix memory leaks or release memory
EError_BufferTooSmall	The supplied buffer is too small. Supply a bigger buffer.
EError_UnableToAllocateMemory	Not enough memory for this allocation.

EError_UnableToAccessImageMemory	Unable to access image memory - Load or size image
EError_RoiTooLarge	ROI too large - Fit ROI to image
EError_NotAValidImage	Not a valid image - Check image contents
EError_ImagesNotSameSize	Images not of the same size - Adjust image size(s)
EError_ImagesNotSameBitsPerPixel	Images not of the same depth (bits per pixel) - Choose compatible types
EError_SourceImageTooSmall	Origin image too small - Use a larger image
EError_PixelsMustHaveFiniteSize	Pixels must have finite size - Use non-zero parameters
EError_ConstantIsNull	Constant is NULL - Use non-zero value
EError_PixelNullEncountered	NULL pixel encountered - Avoid division by zero
EError_ImagesMayNotOverlap	Images cannot overlap - Use distinct images
EError_RoiOutOfImageLimits	ROI out of the top parent limits - Resize to fit in image
EError_RoiAlreadyHasAParent	ROI already has a parent - Detach ROI first
EError_RoiHasNoParentImage	There is no image ancestor for this ROI - Attach the ROI or one of its ancestor to an image
EError_CannotApplyToAnImage	Cannot apply to an image - Apply to a ROI instead
EError_UnsupportedImageType	Unsupported image type - Check type compatibility

EError_InvalidImageType	Invalid image type - Check type compatibility
EError_UnsupportedXserverDepth	Unsupported X server depth
EError_InconsistentRoiHierarchy	The hierarchy of ROI has been corrupted (inconsistent parent/daughters relationship)
EError_SourceImageTooBig	Original image too big - Use a smaller image
EError_BW1RoiNotAligned	First bit index of an aligned ROI must be 0 - Use an aligned ROI
EError_WrongRoiType	Wrong ROI or image type
EError_CyclingParenthodNotAllowed	Cycling Parenthod not allowed
EError_WrongBitsPerRow	Bits per row must be a multiple of 32 (4 bytes) and must be enough to hold all the pixels of an image row.
EError_MisalignedImagePtr	The supplied image pointer must be aligned to 4 bytes.
EError_UnsupportedImageTypeConversion	Unsupported image type conversion
EError_ImageFromFileDoesNotFitIntoROI	The ROI is not the same size as the image file. When loading an image from a file into a ROI, the ROI must have the exact required size. On the other, when loading into an image object, it gets resized to the correct size.
EError_PixelCoordinatesOutOfROI	The specified coordinate is outside the ROI/Image

EError_PixelOutsidePerimeter	Pixel outside perimeter - Check pixel value
EError_PixelInsidePerimeter	Pixel inside perimeter - Check pixel value
EError_IsolatedPixel	Isolated pixel - Check pixel value
EError_MaxPixelInContourReached	Maximum pixels in contour reached
EError_NotAValidContour	Not a valid contour - Initialize using a contouring function
EError_UnableToAccessVectorMemory	Unable to access vector memory - Check proper vector initialization
EError_NotAValidVectorDescriptor	Not a valid vector descriptor
EError_VectorTypeIsNotHist	Vector type is not histogram
EError_NotEnoughGroupsInVector	Not enough groups in vector
EError_InvalidVectorDataSize	Invalid vector data size
EError_InvalidVectorDataType	Invalid vector data type
EError_InvalidVectorType	Invalid vector type
EError_ResultTooBigToFitInVector	Result too big to fit in vector
EError_GroupOutOfRange	Group out of range - Adjust group index
EError_InvalidVectorGroupLength	Invalid vector group length
EError_InvalidNumberOfVectorElements	Invalid number of vector elements - Check proper vector initialization

EError_VectorsNotSameSize	Vectors not of the same size - Adjust vector size(s)
EError_UnableToAccessKernelMemory	Unable to access kernel memory - Check proper kernel initialization
EError_NotAValidKernelDescriptor	Not a valid kernel descriptor
EError_InvalidKernel	Invalid kernel
EError_KernelInvalidSize	Invalid kernel size - Check proper kernel initialization
EError_KernelNotAllocated	Kernel not allocated - Check proper kernel initialization
EError_BadListPosition	Bad list position - Restart list traversal
EError_ListIsEmpty	List is empty
EError_TopOfList	Top of list - Do not traverse backwards
EError_BotOfList	Bottom of list - Do not traverse forwards
EError_ListError	List error
EError_LicenseMissing	The license for this library is not granted - Launch License Manager
EError_EasyImageLicenseMissing	The license for EasyImage is not granted - Launch License Manager
EError_EasyColorLicenseMissing	The license for EasyColor is not granted - Launch License Manager
EError_EasyObjectLicenseMissing	The license for EasyObject is not granted - Launch License Manager

EError_EasyMatchLicenseMissing	The license for EasyMatch is not granted - Launch License Manager
EError_EasyGaugeLicenseMissing	The license for EasyGauge is not granted - Launch License Manager
EError_EasyFindLicenseMissing	The license for EasyFind is not granted - Launch License Manager
EError_EasyOcrLicenseMissing	The license for EasyOCR is not granted - Launch License Manager
EError_EasyOcvLicenseMissing	The license for EasyOCV is not granted - Launch License Manager
EError_EasyBarCodeLicenseMissing	The license for EasyBarCode is not granted - Launch License Manager
EError_EasyMatrixCodeLicenseMissing	The license for EasyMatrixCode is not granted - Launch License Manager
EError_EasyMatchAlignmentModeLicenseMissing	The license for EasyMatch Alignment mode is not granted - Launch License Manager
EError_EvisionStudioLicenseMissing	The license for eVison Studio is not granted - Launch License Manager
EError_CannotWriteOEMKey	The OEM key cannot be set
EError_WarpImagesTooSmall	Warp images too small - Increase image size
EError_UnsupportedImageSize	Unknown error code
EError_UnknownFeature	Unknown feature - Check parameters
EError_InvalidSelectionArgument	Invalid selection argument - Check parameters

EError_SortListTooLong	Sort list too long
EError_NotAValidOperationCode	Not a valid operation code
EError_TooManyObjectsDetected	Too many objects detected - Increase MaxObjects
EError_InvalidFeature	Invalid feature - Check parameters
EError_FeatureNotCalculated	Feature not calculated - Call AnalyseObjects method
EError_BadObjectNumber	Bad object number - Check parameters
EError_NoObjectSelected	No object selected - Blob list is empty
EError_LowThresholdHigherThanHighThreshold	Low threshold higher than high threshold - Adjust thresholds
EError_InvalidThresholdMode	Invalid threshold mode - Use appropriate threshold setting method
EError_NoImageAttached	No image attached to the selection - Use Attach()
EError_OutOfContinuousMode	Invalid call out of continuous mode
EError_InvalidImageTypeForSegmenter	The current segmenter can not cope with this type of image
EError_LayersOverlapping	Two different layers are associated with the same layer index
EError_EndOfIterator	The iterator has reached the end of the enumeration

EError_NoThresholdComputedYet	The threshold valued has not been computed yet - First encode an image
EError_FeatureNotDrawable	This kind of feature cannot be drawn
EError_OnlyApplicableToObjectSelection	This kind of feature cannot be used out of EObjectSelection
EError_MoreThanOneLayerEncoded	Please specify the layer index (several layers are encoded)
EError_CodedElementNotSelected	The coded element is not present in the selection
EError_NoPatternLearnt	No pattern learnt - Load from file or train pattern
EError_PatternTooLarge	Pattern too large - Use a smaller one
EError_PatternTooSmall	Pattern too small - Use a larger one
EError_NotAnEasyMatchFile	Not an EasyMatch file - Check file source
EError_UnsupportedEasyMatchFileVersion	Unsupported EasyMatch file version - Upgrade to a newer release
EError_NoImageLearnt	No image learnt - Call LearnImage() first
EError_WrongNumberOfDegreesOfFreedom	The number of degrees of freedom must be at least one, and no more than five - Use a value in this range

EError_InsufficientContrast	
EError_PatternTooCloseToImageBorder	<p>Not enough feature points - Use a more contrasted pattern or reduce the Don't Care mask</p> <p>Pattern is too close to image border - Leave a margin around the pattern</p>
EError_IncompatibleModes	Incompatible modes (CoarseToFineAnalysisMode and PatternType)
EError_AllowancesAndPatterntypeNotCompatible	Angle and Scale allowances can not be used with the current pattern type
EError_ModelNotSuitedForContrastingregions	The model is unsuitable for ContrastingRegions pattern type - Try an other pattern type or increase surface of region(s)
EError_ModelNotSuitedForConsistentedges	The model is unsuitable for ConsistentEdges pattern type - Try another pattern type or increase the model surface
EError_NoPatternsLoaded	No patterns loaded - Load font file or train
EError_NoPatternsInTheseClasses	No patterns in these classes - Check pattern and text class assignments
EError_CharacterTooSmall	Character too small - Enlarge to font size

EError_CharacterCodeTooBig8	Character code too big to fit in a string, use ReadTextWide instead
EError_CharacterCodeTooBig16	Character code too big to fit in a wide string, use GetFirstCharCode instead
EError_InvalidTextStructure	Text parameter doesn't fit the text topology
EError_InvalidFontFile	The specified font file couldn't be loaded
EError_InvalidTopology	The specified topology is invalid
EError_InvalidEOCR2File	The file-type and structure could not be verified
EError_EOCR2CharCodeNotSet	Character code not set
EError_MismatchingColorSystem	Mismatching color system - Check transform compatibility
EError_ColordLookupMustBeInitialized	Color lookup must be initialized - Use initialization method
EError_UnsupportedColorTransform	Unsupported color transform
EError_UnknownSymbolSize	Unknown symbol size - Check size initialization
EError_UnknownEccFamily	Unknown ECC family (ECC 000/050/080/100/140/200 only)
EError_UncorrectableErrors	

EError_CouldNotLocateSymbol Could not locate the dot matrix symbol (no good candidate object) - See Reference	Too many errors, cannot correct contents - See Reference
EError_UnknownFormatId	Unknown Format ID in ECC 000-140 symbol (Base 11/27/41/37 and ASCII 7/8 only) - See Reference
EError_InvalidCrc	Invalid CRC after error correction in ECC 000-140 symbol - See Reference
EError_CouldNotDecodeSymbol	Could not decode symbol - Try to improve image quality
EError_CouldNotGrade	Could not grade symbol - Quiet zone out of bounds
EError_CouldNotLocateBarcode	Could not locate bar code symbol - Improve contrast, avoid clutter
EError_UnrecognizedSignature	Unrecognized signature - Check enabled symbologies
EError_InvalidNumberOfBars	Invalid number of bars - Improve bar/space contrast
EError_ExtraEdgesFound	Extra edges found - Improve bar/space contrast or uniformity
EError_IncoherentBarSpaceThickness	Incoherent bar/space thickness sequence - Check enabled symbologies
EError_InvalidCheckCharacter	Invalid checksum character - Check enabled symbologies
EError_SymbologyNotEnabled	Symbology not enabled - Invoke method SetSymbologies()

EError_NoEdgesFound	No edges found - Adjust location or improve bar/space contrast
EError_NotAnEasyOcvFile	Not an EasyOCV file - Check file source
EError_UnsupportedEasyOcvFileVersion	Unsupported EasyOCV file version - Upgrade Open eVision
EError_NotEnoughSampleImages	Not enough sample images - Use AddToStatistics
EError_NotAnEcheckerFile	Not an EChecker file - Check file source
EError_UnsupportedEcheckerFileVersion	Unsupported EChecker file version - Upgrade Open eVision
EError_NotEnoughSamplesLearnt	Not enough samples learnt - Use UpdateStatistics
EError_InvalidNormalizationMode	Invalid normalization mode - Check SetNormalize call
EError_ImageNotRegistered	Image not registered - Use method Register before Learn
EError_InvalidLearningSequence	Invalid learning sequence - Use AVERAGE followed by ABS_DEVIATION, or RMS_DEVIATION, then READY
EError_E_ERROR_CONTRAST_TOO_LOW	Image contrast is too low
EError_MotherAlreadyHasThisDaughter	Mother already has this daughter - Detach daughter first
EError_ShapeAlreadyHasDaughters	Shape already has daughters - Detach daughters first
EError_NoValidPointFound	

Not in list attachment mode - Detach daughters first	EError_NotInListAttachmentMode No valid point found in the transition computation.
EError_NotInIndexedAttachmentMode	Not in indexed attachment mode - Detach daughters and call SetIndexed first
EError_UnsupportedShapeVersion	Unsupported shape version - Upgrade Open eVision
EError_RawCalibrationMode	Raw calibration mode - Cannot be used for this operation
EError_BadLandmarkLayout	The layout of supplied landmarks makes the calibration impossible - Check landmarks positions
EError_IncompatibleCalibrationModes	Incompatible calibration modes - Check calibration mode categories
EError_NotEnoughLandmarks	Not enough landmarks to calibrate - Add landmarks or check calibration mode categories
EError_UnexpectedShapeTypeInFile	Unexpected shape type in file - Check target shape against file model root
EError_UnsupportedModelFileVersion	Unsupported model file version - Upgrade Open eVision
EError_CannotAttachDetachWorldShapes	Cannot Attach or Detach World shape - World shapes never have a mother
EError_UnexpectedWorldShapeInFile	Unexpected World Shape in file - Check target shape against file model root

EError_UnexpectedFrameShapeInFile	Unexpected Frame Shape in file - Check target shape against file model root
EError_UnexpectedPointShapeInFile	Unexpected Point Shape in file - Check target shape against file model root
EError_UnexpectedLineShapeInFile	Unexpected Line Shape in file - Check target shape against file model root
EError_UnexpectedCircleShapeInFile	Unexpected Circle Shape in file - Check target shape against file model root
EError_UnexpectedRectangleShapeInFile	Unexpected Rectangle Shape in file - Check target shape against file model root
EError_UnexpectedWedgeShapeInFile	Unexpected Wedge Shape in file - Check target shape against file model root
EError_UnexpectedPointGaugeInFile	Unexpected Point Gauge in file - Check target shape against file model root
EError_UnexpectedLineGaugeInFile	Unexpected Line Gauge in file - Check target shape against file model root
EError_UnexpectedCircleGaugeInFile	Unexpected Circle Gauge in file - Check target shape against file model root
EError_UnexpectedRectangleGaugeInFile	Unexpected Rectangle Gauge in file - Check target shape against file model root
EError_UnexpectedWedgeGaugeInFile	Unexpected Wedge Gauge in file - Check target shape against file model root

EError_UnexpectedBarCodeInFile	Unexpected Bar Code model in file - Check target shape against file model root
EError_AnActiveCurvedEdgeIsRequired	At least one curved edge must be active - Activate r and/or R edges
EError_BrokenWedgeShapeConstraints	Constraints between the geometric and the tolerance of the wedge are broken
EError_FlexnetHandleInitializationFailed	Licensing handle initialization failed
EError_FlexnetLoadingActivationLibraryFailed	Loading of the activation library failed
EError_FlexnetInitializationActivationLibraryFailed	Initialization of activation library failed
EError_FlexnetActivationLibraryMismatch	Activation library mismatch
EError_FlexnetActivationLibraryUnloaded	Activation library component has been unloaded
EError_FlexnetLicensingServiceNotInstalled	The licensing service is not installed
EError_FlexnetNotEnoughRights	Not enough rights to talk to service
EError_FlexnetLicenseJobCreationFailed	License job creation failed
EError_FLEXnetLicensePromptForFileFailed	Unable to disable license finder dialog

EError_PixelHandling	Internal error during image processing
EError_EmptyMorphologicalKernel	Use of a morphological kernel without any element set
EError_MatrixOperation	Internal error during matrix processing
EError_NonSquareMatrix	The operation is only valid for square matrices
EError_IncompatibleMatrixSizes	The sizes of the matrix are incompatible for the operation
EError_UnderdeterminedMatrix	Unsupported operation: The matrix has less rows than columns
EError_OverdeterminedMatrix	Unsupported operation: The matrix has more rows than columns
EError_PointAtInfinity	Unable to apply this operation to points at infinity
EError_NotEnoughCalibrationPoints	Not enough points for the calibration process to succeed
EError_LineAtInfinity	Unable to apply this operation to lines at infinity
EError_UndeterminedGeometricEntity	Undetermined geometric entity in projective geometry
EError_InternalError_000	Internal error 0
EError_InternalError_001	Internal error 1

EError_InternalError_002	Internal error 2
EError_InternalError_003	Internal error 3
EError_InternalError_004	Internal error 4
EError_InternalError_005	Internal error 5
EError_InternalError_006	Internal error 6
EError_InternalError_007	Internal error 7
EError_InternalError_008	Internal error 8
EError_InternalError_009	Internal error 9
EError_InternalError_010	Internal error 10
EError_InternalError_011	Internal error 11
EError_InternalError_012	Internal error 12
EError_InternalError_013	Internal error 13
EError_InternalError_014	Internal error 14
EError_InternalError_015	Internal error 15
EError_InternalError_016	Internal error 16
EError_InternalError_017	Internal error 17
EError_InternalError_018	Internal error 18

EError_InternalError_019	Internal error 19
EError_InternalError_020	Internal error 20
EError_InternalError_021	Internal error 21
EError_InternalError_022	Internal error 22
EError_InternalError_023	Internal error 23
EError_InternalError_024	Internal error 24
EError_InternalError_025	Internal error 25
EError_InternalError_026	Internal error 26
EError_InternalError_027	Internal error 27
EError_InternalError_028	Internal error 28
EError_InternalError_029	Internal error 29
EError_InternalError_030	Internal error 30
EError_InternalError_031	Internal error 31
EError_InternalError_032	Internal error 32
EError_InternalError_033	Internal error 33
EError_InternalError_034	Internal error 34
EError_InternalError_035	Internal error 35

EError_InternalError_036	Internal error 36
EError_InternalError_037	Internal error 37
EError_InternalError_038	Internal error 38
EError_InternalError_039	Internal error 39
EError_InternalError_040	Internal error 40
EError_InternalError_041	Internal error 41
EError_InternalError_042	Internal error 42
EError_InternalError_043	Internal error 43
EError_InternalError_044	Internal error 44
EError_InternalError_045	Internal error 45
EError_InternalError_046	Internal error 46
EError_InternalError_047	Internal error 47
EError_InternalError_048	Internal error 48
EError_InternalError_049	Internal error 49
EError_InternalError_050	Internal error 50
EError_InternalError_051	Internal error 51
EError_InternalError_052	Internal error 52

EError_InternalError_053	Internal error 53
EError_InternalError_054	Internal error 54
EError_InternalError_055	Internal error 55
EError_InternalError_056	Internal error 56
EError_InternalError_057	Internal error 57
EError_InternalError_058	Internal error 58
EError_InternalError_059	Internal error 59
EError_InternalError_060	Internal error 60
EError_InternalError_061	Internal error 61
EError_InternalError_062	Internal error 62
EError_InternalError_063	Internal error 63
EError_InternalError_064	Internal error 64
EError_InternalError_065	Internal error 65
EError_InternalError_066	Internal error 66
EError_InternalError_067	Internal error 67
EError_InternalError_068	Internal error 68
EError_InternalError_069	Internal error 69

EError_InternalError_070	Internal error 70
EError_InternalError_071	Internal error 71
EError_InternalError_072	Internal error 72
EError_InternalError_073	Internal error 73
EError_InternalError_074	Internal error 74
EError_InternalError_075	Internal error 75
EError_InternalError_076	Internal error 76
EError_InternalError_077	Internal error 77
EError_InternalError_078	Internal error 78
EError_InternalError_079	Internal error 79
EError_InternalError_080	Internal error 80
EError_InternalError_081	Internal error 81
EError_InternalError_082	Internal error 82
EError_InternalError_083	Internal error 83
EError_InternalError_084	Internal error 84
EError_InternalError_085	Internal error 85
EError_InternalError_086	Internal error 86

EError_InternalError_087	Internal error 87
EError_InternalError_088	Internal error 88
EError_InternalError_089	Internal error 89
EError_InternalError_090	Internal error 90
EError_InternalError_091	Internal error 91
EError_InternalError_092	Internal error 92
EError_InternalError_093	Internal error 93
EError_InternalError_094	Internal error 94
EError_InternalError_095	Internal error 95
EError_InternalError_096	Internal error 96
EError_InternalError_097	Internal error 97
EError_InternalError_098	Internal error 98
EError_InternalError_099	Internal error 99
EError_InternalError_100	Internal error 100
EError_CannotTraceErrors	Cannot trace errors because of a system failure
EError_NotImplemented	Feature not implemented
EError_NullPointer	The supplied pointer is NULL

EError_InvalidTimeout	The current timeout value is 0
EError_InvalidTimeoutReentrancy	Cannot Stop a timeout that has not been started. Cannot Pop a timeout that has not been pushed
EError_InvalidTimeoutState	Cannot Start a timeout that has been reached Cannot Pop a timeout that is Active
EError_Unknown	Unknown error

EFamily Enum

Allowed values for the ECC symbol family in EasyMatrixCode.

EFamily_ECC000	ECC 000, no error recovery capability by convolutional coding.
EFamily_ECC050	ECC 050, 2.8 % error recovery capability by convolutional coding.
EFamily_ECC080	ECC 080, 5.5 % error recovery capability by convolutional coding.
EFamily_ECC100	ECC 100, 12.6 % error recovery capability by convolutional coding.
EFamily_ECC140	ECC 140, 25 % error recovery capability by convolutional coding.
EFamily_ECC200	ECC 200, 20 % error recovery capability.
EFamily_Unknown	-

Remarks

EFeature Enum

The various features that can be measured on the coded elements of a selection.

EFeature_ElementIndex	Index of the coded element (cf. ECodedElement::ElementIndex).
EFeature_LayerIndex	Index of the layer of the coded element (cf. ECodedElement::LayerIndex).
EFeature_RunCount	Number of runs (cf. ECodedElement::RunCount).
EFeature_Area	Number of pixels (cf. ECodedElement::Area).
EFeature_LargestRun	Length of the largest run (cf. ECodedElement::LargestRun).
EFeature_ContourX	Starting point abscissa of the contour of the coded element (cf. ECodedElement::ContourX).
EFeature_ContourY	Starting point ordinate of the countour of the coded element (cf. ECodedElement::ContourY).
EFeature_LeftLimit	Abscissa of the leftmost pixel (cf. ECodedElement::LeftLimit).
EFeature_RightLimit	Abscissa of the rightmost pixel (cf. ECodedElement::RightLimit).
EFeature_TopLimit	Abscissa of the topmost pixel (cf. ECodedElement::TopLimit).

EFeature_BottomLimit	Ordinate of the bottommost pixel (cf. ECodedElement::BottomLimit).
EFeature_GravityCenterX	Abscissa of the gravity center (cf. ECodedElement::GravityCenterX).
EFeature_GravityCenterY	Ordinate of the gravity center (cf. ECodedElement::GravityCenterY).
EFeature_BoundingBoxCenterX	Abscissa of the center of the bounding box (cf. ECodedElement::BoundingBoxCenterX).
EFeature_BoundingBoxCenterY	Ordinate of the center of the bounding box (cf. ECodedElement::BoundingBoxCenterY).
EFeature_BoundingBoxWidth	Width of the bounding box (Feret diameter 0° - cf. ECodedElement::BoundingBoxWidth).
EFeature_BoundingBoxHeight	Height of the bounding box (Feret diameter 90° - cf. ECodedElement::BoundingBoxHeight).
EFeature_FeretBox22CenterX	Abscissa of the center of the Feret box oriented at 22.5° (cf. ECodedElement::FeretBox22CenterX).
EFeature_FeretBox22CenterY	Ordinate of the center of the Feret box oriented at 22.5° (cf. ECodedElement::FeretBox22CenterY).
EFeature_FeretBox22Width	Width of the Feret box oriented at 22.5° (Feret diameter at 22.5° - cf. ECodedElement::FeretBox22Width).
EFeature_FeretBox22Height	Height of the Feret box oriented at 22.5° (Feret diameter at 112.5° - cf. ECodedElement::FeretBox22Height).

EFeature_FeretBox45CenterX	Abscissa of the center of the Feret box oriented at 45° (cf. ECodedElement::FeretBox45CenterX).
EFeature_FeretBox45CenterY	Ordinate of the center of the Feret box oriented at 45° (cf. ECodedElement::FeretBox45CenterY).
EFeature_FeretBox45Width	Width of the Feret box oriented at 45° bounding box (Feret diameter at 45° - cf. ECodedElement::FeretBox45Width).
EFeature_FeretBox45Height	Height of the Feret box oriented at 45° (Feret diameter at 135° - cf. ECodedElement::FeretBox45Height).
EFeature_FeretBox68CenterX	Abscissa of the center of the Feret box oriented at 67.5° (cf. ECodedElement::FeretBox68CenterX).
EFeature_FeretBox68CenterY	Ordinate of the center of the Feret box oriented at 67.5° (cf. ECodedElement::FeretBox68CenterY).
EFeature_FeretBox68Width	Width of the Feret box oriented at 67.5° (Feret diameter at 67.5° - cf. ECodedElement::FeretBox68Width).
EFeature_FeretBox68Height	Height of the Feret box oriented at 67.5° (Feret diameter at 157.5° - cf. ECodedElement::FeretBox68Height).
EFeature_MinimumEnclosingRectangleCenterX	Abscissa of the Minimum Enclosing Rectangle center (cf. ECodedElement::MinimumEnclosingRectangleCenterX).
EFeature_MinimumEnclosingRectangleCenterY	Ordinate of the Minimum Enclosing Rectangle center (cf. ECodedElement::MinimumEnclosingRectangleCenterY).

EFeature_MinimumEnclosingRectangleWidth	Width of the Minimum Enclosing Rectangle (cf. ECodedElement::MinimumEnclosingRectangleWidth).
EFeature_MinimumEnclosingRectangleHeight	Height of the Minimum Enclosing Rectangle (cf. ECodedElement::MinimumEnclosingRectangleHeight).
EFeature_MinimumEnclosingRectangleAngle	Direction of the Minimum Enclosing Rectangle (cf. ECodedElement::MinimumEnclosingRectangleAngle).
EFeature_SigmaX	Centered moment of inertia around X (average squared X-deviation - cf. ECodedElement::SigmaX).
EFeature_SigmaY	Centered moment of inertia around Y (average squared Y-deviation - cf. ECodedElement::SigmaY).
EFeature_SigmaXX	Reduced, centered moment of inertia (around the principal inertia axis - cf. ECodedElement::SigmaXX).
EFeature_SigmaXY	Centered cross moment of inertia (average X-deviation * Y-deviation - cf. ECodedElement::SigmaXY).
EFeature_SigmaYY	Reduced, centered moment of inertia (around the secondary inertia axis - cf. ECodedElement::SigmaYY).
EFeature_EllipseWidth	Long axis of the ellipse of inertia (cf. ECodedElement::EllipseWidth).
EFeature_EllipseHeight	Short axis of the ellipse of inertia (cf. ECodedElement::EllipseHeight).

EFeature_EllipseAngle	Angle of the principal axis of the ellipse of inertia (cf. ECodedElement::EllipseAngle).
EFeature_Eccentricity	Eccentricity of the ellipse of inertia (cf. ECodedElement::Eccentricity).
EFeature_FeretBoxCenterX	Abscissa of the center of the Feret box oriented at a fixed angle (cf. ECodedElement::ComputeFeretBox). The angle of interest is set through EObjectSelection::FeretAngle .
EFeature_FeretBoxCenterY	Ordinate of the center of the Feret box oriented at a fixed angle (cf. ECodedElement::ComputeFeretBox). The angle of interest is set through EObjectSelection::FeretAngle .
EFeature_FeretBoxWidth	Width of the Feret box oriented at a fixed angle (cf. ECodedElement::ComputeFeretBox). The angle of interest is set through EObjectSelection::FeretAngle .
EFeature_FeretBoxHeight	Height of the Feret box oriented at a fixed angle (cf. ECodedElement::ComputeFeretBox). The angle of interest is set through EObjectSelection::FeretAngle .
EFeature_PixelMin	Minimum gray level of the pixels of the attached image over the coded element (cf. ECodedElement::ComputePixelMin). The attached image is set through EObjectSelection::AttachedImage .
EFeature_PixelMax	Maximum gray level of the pixels of the attached image over the coded element (cf. ECodedElement::ComputePixelMax). The attached image is set through EObjectSelection::AttachedImage .

<p>EFeature_WeightedGravityCenterX</p> <p>EFeature_WeightedGravityCenterY</p> <p>Ordinate of the gravity center of the pixels of the attached image over the coded element (cf. ECodedElement::ComputeWeightedGravityCenter). The attached image is set through EObjectSelection::AttachedImage.</p>	<p>Abscissa of the gravity center of the pixels of the attached image over the coded element (cf. ECodedElement::ComputeWeightedGravityCenter). The attached image is set through EObjectSelection::AttachedImage.</p>
<p>EFeature_PixelGrayAverage</p>	<p>Average gray-level value of the attached image over the coded element (cf. ECodedElement::ComputePixelGrayAverage). The attached image is set through EObjectSelection::AttachedImage.</p>
<p>EFeature_PixelGrayVariance</p>	<p>Variance of the gray-level value of the attached image over the coded element (cf. ECodedElement::ComputePixelGrayVariance). The attached image is set through EObjectSelection::AttachedImage.</p>
<p>EFeature_PixelGrayDeviation</p>	<p>Standard deviation of the gray-level value of the attached image over the coded element (cf. ECodedElement::ComputePixelGrayDeviation). The attached image is set through EObjectSelection::AttachedImage.</p>

EFiltreringMode Enum

Allowed values for the filtering mode of EasyMatch.

<p>EFiltreringMode_Uniform</p>	<p>Filtering with a uniform 2x2 kernel. This is the preferred mode for natural images. Default mode.</p>

EFilteringMode_LowPass

Filtering with a low-pass 3x3 kernel. This is the preferred mode for images featuring sharp gray-level transitions.

F

indContrastMode Enum

Allowed values for the contrast mode of EasyMatch.

EFindContrastMode_Normal	Accepts instances with normal contrast (default mode).
EFindContrastMode_Inverse	Accepts instances with reversed contrast.
EFindContrastMode_Any	Accepts instances with normal and/or reversed contrast.
EFindContrastMode_PointByPointNormal	-
EFindContrastMode_PointByPointInverse	-
EFindContrastMode_PointByPointAny	-
EFindContrastMode_Unknown	-

EFlipping Enum

Allowed values for the symbol flipping type in EasyMatrixCode.

EFlipping_Yes	Image is flipped.
EFlipping_No	Image is not flipped.
EFlipping_Unknown	To be determined at Read or Learn time.

EFramePosition Enum

This enumeration contains the possible values for the placement of the overlay frame edges that are drawn to highlight the position of an ROI.

EFramePosition_On	The frame is centered on the ROI edges.
EFramePosition_Inside	The outer edges of the frame remain totally inside the ROI.
EFramePosition_Outside	The inner edges of the frame remain totally outside the ROI.

EGrayscaleSingleThreshold Enum

The modes that are available to segment a grayscale image using a single threshold.

EGrayscaleSingleThreshold_Absolute	Thresholds the image against a fixed, absolute gray level. The threshold value is fixed through EGrayscaleSingleThresholdSegmenter::AbsoluteThreshold .

<p><code>EGrayscaleSingleThreshold_Relative</code></p> <p><code>EGrayscaleSingleThreshold_MinResidue</code></p>	<p>Thresholds the image against a relative gray level: The actual threshold is selected so that a given fraction of the pixels of the image lie below it. The fraction is fixed through EGrayscaleSingleThresholdSegmenter::RelativeThreshold.</p>
<p>Thresholds the image using an automatically-computed value such that the quadratic difference between the source and thresholded image is minimized.</p>	<p><code>EGrayscaleSingleThreshold_MaxEntropy</code></p> <p>Thresholds the image using an automatically-computed value such that the entropy (i.e. the amount of information) of the resulting thresholded image is maximized.</p>
<p><code>EGrayscaleSingleThreshold_IsoData</code></p>	<p>Thresholds the image using an automatically-computed value halfway between the average dark gray value (i.e. gray levels below the threshold) and average light gray values (i.e. gray levels above the threshold).</p>

EHarrisThresholdingMode Enum

The thresholding modes for the Harris corner detector.

<code>EHarrisThresholdingMode_Relative</code>	Relative thresholding mode.
<code>EHarrisThresholdingMode_Absolute</code>	Absolute thresholding mode.

EHistogramFeature Enum

The various parameters that can be extracted from a histogram.

EHistogramFeature_MostFrequentPixelValue	Value of the most frequent pixel.
EHistogramFeature_MostFrequentPixelFrequency	Frequency of the most frequent pixel.
EHistogramFeature_LeastFrequentPixelValue	Value of the least frequent pixel.
EHistogramFeature_LeastFrequentPixelFrequency	Frequency of the least frequent pixel.
EHistogramFeature_SmallestPixelValue	Smallest pixel value.
EHistogramFeature_GreatestPixelValue	Largest pixel value.
EHistogramFeature_PixelCount	Number of pixels.
EHistogramFeature_AveragePixelValue	Mean of the pixel values.
EHistogramFeature_PixelValueStdDev	Standard deviation of the pixel values.

EHitAndMissValue Enum

The allowed values for the elements of a hit-and-miss kernel.

EHitAndMissValue_Background	The element belongs to the background.
EHitAndMissValue_DontCare	The element does not belongs to the background, neither to the foreground. It is ignored by the kernel.
EHitAndMissValue_Foreground	The element belongs to the foreground.

EImageFileType Enum

-

EImageFileType_Bmp	-
EImageFileType_Jpeg2000	-
EImageFileType_Jpeg	-
EImageFileType_Png	-
EImageFileType_Tiff	-
EImageFileType_Auto	-
EImageFileType_Euresys	-

EImageFileType_Unknown

-
E

|

ImageType Enum

Image type.

EImageType_BW1	Bi-level image.
EImageType_BW8	8 bits per pixel gray-level image.
EImageType_BW16	16 bits per pixel gray-level image
EImageType_BW32	32 bits per pixel gray-level image.
EImageType_C15	15 bits per pixel color image (R:5, G:5, B:5).
EImageType_C16	16 bits per pixel color image (R:5, G:6, B:5).
EImageType_C24	24 bits per pixel color image (RGB).
EImageType_C24A	32 bits per pixel color image (RGB + unused alpha channel).
EImageType_C48	48 bits per pixel color image (RGB).
EImageType_Depth8	8 bits per pixel gray-level image.
EImageType_Depth16	16 bits per pixel gray-level image.
EImageType_Depth32f	32 bits per pixel gray-level image.

Remarks

For example, an [EImageC24](#) has type value [EImageType_C24](#) and its pixels are typed as [EC24](#).

EKernelRectifier Enum

Possible values for the rectification mode of a kernel. This property allows specifying how negative convolution result values are handled.

<code>EKernelRectifier_DoNotRectify</code>	The offset of the kernel is added to the values resulting from the convolution. Negative values are then set to zero, and the values that exceed the maximum value for the image type are set to this maximum value.
<code>EKernelRectifier_KeepNegative</code>	The positive values are discarded (set to zero) and the magnitude (absolute value) of the negative values is used.
<code>EKernelRectifier_KeepPositive</code>	Positive value is used. The negative values are discarded (set to zero).
<code>EKernelRectifier_Absolute</code>	The absolute value is used.

EKernelRotation Enum

Possible values for rotating a convolution kernel.

<code>EKernelRotation_NoRotation</code>	No rotation of the structuring element.

EKernelRotation_Clockwise	Clockwise rotation (one full turn per pass).
EKernelRotation_Anticlockwise	Counterclockwise rotation (one full turn per pass).

EKernelType Enum

The types of convolution kernels that are supported by Open eVision.

EKernelType_WhiteSkelet	White skeleton morphological probe.
EKernelType_BlackSkelet	Black skeleton morphological probe.
EKernelType_Edge	Edge detection morphological probe.
EKernelType_SobelX	X-axis Sobel derivative.
EKernelType_SobelY	Y-axis Sobel derivative.
EKernelType_PrewittX	X-axis Prewitt derivative.
EKernelType_PrewittY	Y-axis Prewitt derivative.
EKernelType_Laplacian4	4-connected Laplacian.
EKernelType_Laplacian8	8-connected Laplacian.
EKernelType_LowPass1	Low pass filter.
EKernelType_LowPass2	Low pass filter (average of neighbors).
EKernelType_LowPass3	Low pass filter (average).

EKernelType_HighPass1	High pass filter (value plus 4-connected Laplacian).
EKernelType_HighPass2	High pass filter (value plus 8-connected Laplacian).
EKernelType_Sobel	-
EKernelType_Prewitt	-
EKernelType_Roberts	-
EKernelType_Uniform3x3	-
EKernelType_Gaussian3x3	-
EKernelType_Uniform5x5	-
EKernelType_Gaussian5x5	-
EKernelType_Gaussian7x7	-
EKernelType_Uniform7x7	-
EKernelType_LaplacianX	-
EKernelType_LaplacianY	-
EKernelType_Gradient	-
EKernelType_GradientX	-

EKernelType_GradientY	-
EKernelType_Uniform	-
EKernelType_Gaussian	-

ELearningMode Enum

Allowed values for the learning mode in EasyOCV.

ELearningMode_Reset	Restart learning.
ELearningMode_Template	Train the mother image.
ELearningMode_Average	Accumulate an image for average estimation.
ELearningMode_RmsDeviation	Accumulate an image for standard deviation estimation.
ELearningMode_AbsDeviation	Accumulate an image for robust deviation estimation.
ELearningMode_Ready	End learning and compute threshold images.

ELearnParam Enum

Allowed values for the kind of parameters that can be learnt by EasyMatrixCode.

ELearnParam_LogicalSize

The data matrix code contrast types the candidate is matched against at read time.	ELearnParam_ContrastType The data matrix code symbol logical sizes the candidate is matched against at read time.
ELearnParam_Flipping	The data matrix code flipping values the candidate is matched against at read time.
ELearnParam_Family	The data matrix code families the candidate is matched against at read time.
ELearnParam_NumItems	-

ELegacyFeature Enum

The various parameters that can be extracted from a histogram. This enumeration pertains to the EasyObject legacy API. Please use [ECodedImage2](#) instead.

ELegacyFeature_NoFeature	-
ELegacyFeature_Class	Class number.
ELegacyFeature_RunsNumber	Number of runs.
ELegacyFeature_Area	Number of pixels. (<i>Signed Integer</i>).
ELegacyFeature_LargestRun	Size of the longest run. (<i>Signed Integer</i>).
ELegacyFeature_GravityCenterX	Abscissa of the gravity center. (<i>Float</i>). (*)

ELegacyFeature_GravityCenterY	Ordinate of the gravity center. (<i>Float</i>). (*)
ELegacyFeature_LimitCenterX	Abscissa of the center of the bounding box. (<i>Float</i>). (*)
ELegacyFeature_LimitCenterY	Ordinate of the center of the bounding box. (<i>Float</i>). (*)
ELegacyFeature_LimitWidth	Width of the bounding box (Feret's diameter 0°). (<i>Float</i>). (*)
ELegacyFeature_LimitHeight	Height of the bounding box (Feret's diameter 90°). (<i>Float</i>). (*)
ELegacyFeature_Limit45CenterX	Abscissa of the center of the 45° bounding box. (<i>Float</i>). (*)
ELegacyFeature_Limit45CenterY	Ordinate of the center of the 45° bounding box. (<i>Float</i>). (*)
ELegacyFeature_Limit45Width	Width of the 45° bounding box (Feret's diameter 45°). (<i>Float</i>). (*)
ELegacyFeature_Limit45Height	Height of the 45° bounding box (Feret's diameter 135°). (<i>Float</i>). (*)
ELegacyFeature_ContourX	Starting point abscissa of the object contour. (<i>Signed Integer</i>).
ELegacyFeature_ContourY	Starting point ordinate of the object contour. (<i>Signed Integer</i>).
ELegacyFeature_PixelMin	Minimum gray level of all pixels. (<i>Signed Integer</i>).
ELegacyFeature_PixelMax	Maximum gray level of all pixels. (<i>Signed Integer</i>).
ELegacyFeature_SigmaX	

ELegacyFeature_SigmaY Centered moment of inertia around Y (average squared Y-deviation). (<i>Float</i>).	Centered moment of inertia around X (average squared X-deviation). (<i>Float</i>).
ELegacyFeature_SigmaXY	Centered cross moment of inertia (average X-deviation * Y-deviation). (<i>Float</i>).
ELegacyFeature_SigmaXX	Reduced, centered moment of inertia (around the principal inertia axis). (<i>Float</i>).
ELegacyFeature_SigmaYY	Reduced, centered moment of inertia (around the secondary inertia axis). (<i>Float</i>).
ELegacyFeature_EllipseWidth	Long axis of the ellipse of inertia. (<i>Float</i>). (*)
ELegacyFeature_EllipseHeight	Short axis of the ellipse of inertia. (<i>Float</i>). (*)
ELegacyFeature_EllipseAngle	Direction of the principal axis of inertia. (<i>Float</i>). (*)
ELegacyFeature_CentroidX	Abscissa of the weighted gravity center. (<i>Float</i>). (*)
ELegacyFeature_CentroidY	Ordinate of the weighted gravity center. (<i>Float</i>). (*)
ELegacyFeature_PixelGrayAverage	Average gray-level value of the object pixels. (<i>Float</i>).
ELegacyFeature_PixelGrayVariance	Variance of the gray-level value of the object pixels. (<i>Float</i>).
ELegacyFeature_Limit22CenterX	Abscissa of the center of the 22.5° bounding box. (<i>Float</i>). (*)
ELegacyFeature_Limit22CenterY	Ordinate of the center of the 22.5° bounding box. (<i>Float</i>). (*)

ELegacyFeature_Limit22Width	Width of the 22.5° bounding box (Feret's diameter 22.5°). (<i>Float</i>). (*)
ELegacyFeature_Limit22Height	Height the 22.5° bounding box (Feret's diameter 112.5°). (<i>Float</i>). (*)
ELegacyFeature_Limit68CenterX	Abscissa of the center of the 67.5° bounding box. (<i>Float</i>). (*)
ELegacyFeature_Limit68CenterY	Ordinate of the center of the 67.5° bounding box. (<i>Float</i>). (*)
ELegacyFeature_Limit68Width	Width of the 67.5° bounding box (Feret's diameter 67.5°). (<i>Float</i>). (*)
ELegacyFeature_Limit68Height	Height of the 67.5° bounding box (Feret's diameter 157.5°). (<i>Float</i>). (*)
ELegacyFeature_LimitAngledCenterX	Abscissa of the center of the bounding box having a skew angle defined by the LimitAngle property. (<i>Float</i>).
ELegacyFeature_LimitAngledCenterY	Ordinate of the center of the bounding box having a skew angle defined by the LimitAngle property. (<i>Float</i>).
ELegacyFeature_LimitAngledWidth	Width of the bounding box having a skew angle defined by the LimitAngle property (Feret's diameter [LimitAngle]). (<i>Float</i>).
ELegacyFeature_LimitAngledHeight	Height of the bounding box having a skew angle defined by the LimitAngle property (Feret's diameter [LimitAngle + 90°]). (<i>Float</i>).
ELegacyFeature_FeretCenterX	Abscissa of the Feret's bounding box center. (<i>Float</i>). (*)
ELegacyFeature_FeretCenterY	

ELegacyFeature_FeretWidth Width of the Feret's bounding box. (<i>Float</i>). (*)	Ordinate of the Feret's bounding box center. (<i>Float</i>). (*)
ELegacyFeature_FeretHeight	Height of the Feret's bounding box. (<i>Float</i>). (*)
ELegacyFeature_FeretAngle	Direction of the Feret's bounding box. (<i>Float</i>). (*)
ELegacyFeature_ObjectNumber	Identification number.
ELegacyFeature_GravityCenter	Abscissa of the gravity center. (<i>Float</i>). (*)
ELegacyFeature_Limit	Abscissa of the center of the bounding box. (<i>Float</i>). (*)
ELegacyFeature_Limit22	Abscissa of the center of the 22.5° bounding box. (<i>Float</i>). (*)
ELegacyFeature_Limit45	Abscissa of the center of the 45° bounding box. (<i>Float</i>). (*)
ELegacyFeature_Limit68	Abscissa of the center of the 67.5° bounding box. (<i>Float</i>). (*)
ELegacyFeature_LimitAngled	Abscissa of the center of the bounding box having a skew angle defined by the LimitAngle property. (<i>Float</i>).
ELegacyFeature_Ellipse	Long axis of the ellipse of inertia. (<i>Float</i>). (*)
ELegacyFeature_Centroid	-
ELegacyFeature_Feret	-

ELocalSearchMode Enum

Allowed values for the local search mode of EasyFind.

ELocalSearchMode_Basic	Default local search neighborhood. Sets EPatternFinder::AngleSearchExtent , EPatternFinder::ScaleSearchExtent , EPatternFinder::XSearchExtent and EPatternFinder::YSearchExtent to 3.
ELocalSearchMode_ExtendedTranslation	Local search neighborhood extended on the translation degrees of freedom. Sets EPatternFinder::AngleSearchExtent and EPatternFinder::ScaleSearchExtent to 3. Sets EPatternFinder::XSearchExtent and EPatternFinder::YSearchExtent to 5.
ELocalSearchMode_ExtendedAll	Local search neighborhood extended on all degrees of freedom. Sets EPatternFinder::AngleSearchExtent , EPatternFinder::ScaleSearchExtent , EPatternFinder::XSearchExtent and EPatternFinder::YSearchExtent to 5.
ELocalSearchMode_ExtendedMore	Local search neighborhood even more extended on all degrees of freedom. Sets EPatternFinder::AngleSearchExtent and EPatternFinder::ScaleSearchExtent to 7. Sets EPatternFinder::XSearchExtent and EPatternFinder::YSearchExtent to 9.
ELocalSearchMode_Reserved	Reserved for internal use.
ELocalSearchMode_Custom	

ELocationMode Enum

Custom local search neighborhood. Set [EPatternFinder::AngleSearchExtent](#), [EPatternFinder::ScaleSearchExtent](#), [EPatternFinder::XSearchExtent](#) and [EPatternFinder::YSearchExtent](#) to custom values.

Allowed values for the kind of pre-processing to be applied to the sample image in EasyOCV.

ELocationMode_Raw	The raw gray-level image is used to locate the characters.
ELocationMode_Binarized	The gray-level image is thresholded before location, thus enhancing contrast between the characters and the foreground.
ELocationMode_Gradient	The gradient (edge-detection) of the image is operated on.
ELocationMode_Laplacian	The Laplacian of the image is operated on.

Remarks

Experience reveals that the best location reliability is achieved by the [ELocationMode_Binarized](#) and [ELocationMode_Gradient](#) modes. Additionally, the [ELocationMode_Gradient](#) mode is not sensitive to the choice of a threshold level. Use of the [ELocationMode_Laplacian](#) mode is not recommended.

ELogicalSize Enum

Allowed values for the logical size of Data Matrix codes in EasyMatrixCode.

ELogicalSize_9x9	ECC 000-140 squares.
------------------	----------------------

ELogicalSize_11x11	ECC 000-140 squares.
ELogicalSize_13x13	ECC 000-140 squares.
ELogicalSize_15x15	ECC 000-140 squares.
ELogicalSize_17x17	ECC 000-140 squares.
ELogicalSize_19x19	ECC 000-140 squares.
ELogicalSize_21x21	ECC 000-140 squares.
ELogicalSize_23x23	ECC 000-140 squares.
ELogicalSize_25x25	ECC 000-140 squares.
ELogicalSize_27x27	ECC 000-140 squares.
ELogicalSize_29x29	ECC 000-140 squares.
ELogicalSize_31x31	ECC 000-140 squares.
ELogicalSize_33x33	ECC 000-140 squares.
ELogicalSize_35x35	ECC 000-140 squares.
ELogicalSize_37x37	ECC 000-140 squares.
ELogicalSize_39x39	ECC 000-140 squares.
ELogicalSize_41x41	ECC 000-140 squares.
ELogicalSize_43x43	ECC 000-140 squares.

ELogicalSize_45x45	ECC 000-140 squares.
ELogicalSize_47x47	ECC 000-140 squares.
ELogicalSize_49x49	ECC 000-140 squares.
ELogicalSize_10x10	ECC 200 squares.
ELogicalSize_12x12	ECC 200 squares.
ELogicalSize_14x14	ECC 200 squares.
ELogicalSize_16x16	ECC 200 squares.
ELogicalSize_18x18	ECC 200 squares.
ELogicalSize_20x20	ECC 200 squares.
ELogicalSize_22x22	ECC 200 squares.
ELogicalSize_24x24	ECC 200 squares.
ELogicalSize_26x26	ECC 200 squares.
ELogicalSize_32x32	ECC 200 squares.
ELogicalSize_36x36	ECC 200 squares.
ELogicalSize_40x40	ECC 200 squares.
ELogicalSize_44x44	ECC 200 squares.
ELogicalSize_48x48	ECC 200 squares.

ELogicalSize_52x52	ECC 200 squares.
ELogicalSize_64x64	ECC 200 squares.
ELogicalSize_72x72	ECC 200 squares.
ELogicalSize_80x80	ECC 200 squares.
ELogicalSize_88x88	ECC 200 squares.
ELogicalSize_96x96	ECC 200 squares.
ELogicalSize_104x104	ECC 200 squares.
ELogicalSize_120x120	ECC 200 squares.
ELogicalSize_132x132	ECC 200 squares.
ELogicalSize_144x144	ECC 200 squares.
ELogicalSize_8x18	ECC 200 rectangles
ELogicalSize_8x32	ECC 200 rectangles
ELogicalSize_12x26	ECC 200 rectangles
ELogicalSize_12x36	ECC 200 rectangles
ELogicalSize_16x36	ECC 200 rectangles
ELogicalSize_16x48	ECC 200 rectangles
ELogicalSize_Unknown	To be determined at Read or Learn time.

Remarks

EMatchContrastMode Enum

Allowed values for the contrast mode of EasyMatch.

EMatchContrastMode_Normal	Normal contrast. Pattern occurrences will be found with the same contrast as at learn time. Default mode.
EMatchContrastMode_Inverse	Inverse contrast. Pattern occurrences will be found with reversed contrast.
EMatchContrastMode_Any	Normal or inverse contrast. Pattern occurrences can be found both with normal and inverse contrast.

EMatchingMode Enum

Allowed values for the matching mode of EasyOCR.

EMatchingMode_Rms	Root-mean-square error method is used.
EMatchingMode_Standard	Gray-level correlation method is used.
EMatchingMode_Normalized	Normalized gray-level correlation method is used.

EMatrixCodeContrastMode Enum

-	
EMatrixCodeContrastMode_BlackOnWhite	Dark cells on a light background.
EMatrixCodeContrastMode_WhiteOnBlack	Light cells on a dark background.

EMaximumAnalysisMode Enum

This enumeration contains the possible values for the analysis mode of the [ELaserLineExtractor](#) object.

EMaximumAnalysisMode_Peaks	Peak analysis mode.
EMaximumAnalysisMode_Max	Maximum analysis mode.
EMaximumAnalysisMode_COG	Center of gravity analysis mode.

ENormalizationMode Enum

Allowed values for the gray-level normalization mode in EasyOCV.

ENormalizationMode_NoNormalization	No gray-level normalization (contrast changes will be detected).
------------------------------------	--

ENormalizationMode_Moments	Contrast normalization based on linear change.
ENormalizationMode_Threshold	Contrast normalization based on non-linear change.

EOCRClass Enum

Allowed values for the class of pattern in EasyOCR.

EOCRClass__0	Character belongs to class 0.
EOCRClass__1	Character belongs to class 1.
EOCRClass__2	Character belongs to class 2.
EOCRClass__3	Character belongs to class 3.
EOCRClass__4	Character belongs to class 4.
EOCRClass__5	Character belongs to class 5.
EOCRClass__6	Character belongs to class 6.
EOCRClass__7	Character belongs to class 7.
EOCRClass__8	Character belongs to class 8.
EOCRClass__9	Character belongs to class 9.
EOCRClass__10	Character belongs to class 10.

EOCRClass__11	Character belongs to class 11.
EOCRClass__12	Character belongs to class 12.
EOCRClass__13	Character belongs to class 13.
EOCRClass__14	Character belongs to class 14.
EOCRClass__15	Character belongs to class 15.
EOCRClass__16	Character belongs to class 16.
EOCRClass__17	Character belongs to class 17.
EOCRClass__18	Character belongs to class 18.
EOCRClass__19	Character belongs to class 19.
EOCRClass__20	Character belongs to class 20.
EOCRClass__21	Character belongs to class 21.
EOCRClass__22	Character belongs to class 22.
EOCRClass__23	Character belongs to class 23.
EOCRClass__24	Character belongs to class 24.
EOCRClass__25	Character belongs to class 25.
EOCRClass__26	Character belongs to class 26.
EOCRClass__27	Character belongs to class 27.

EOCRClass_28	Character belongs to class 28.
EOCRClass_29	Character belongs to class 29.
EOCRClass_30	Character belongs to class 30.
EOCRClass_Digit	Character belongs to class 0. Equivalent to EOCRClass_0 .
EOCRClass_UpperCase	Character belongs to class 1. Equivalent to EOCRClass_1 .
EOCRClass_LowerCase	Character belongs to class 2. Equivalent to EOCRClass_2 .
EOCRClass_Special	Character belongs to class 3. Equivalent to EOCRClass_3 .
EOCRClass_Extended	Character belongs to class 4. Equivalent to EOCRClass_4 .
EOCRClass_AllClasses	Character belongs to all classes, from 0 to 31 included.

EOCRColor Enum

Allowed values for the text color in EasyOCR.

EOCRColor_BlackOnWhite	The characters appear darker than the background.
EOCRColor_WhiteOnBlack	The characters appear lighter than the background.

EOCRCColor_DarkOnLight	The characters appear darker than the background. No thresholding takes place when the characters are learnt and/or recognized.
------------------------	---

EOCRCColor_LightOnDark	The characters appear lighter than the background. No thresholding takes place when the characters are learnt and/or recognized.
------------------------	--

EPatternType Enum

Allowed values for the type of patterns in EasyFind.

EPatternType_ConsistentEdges	Defines the ConsistentEdges pattern type.
EPatternType_ContrastingRegions	Defines the ContrastingRegions pattern type.
EPatternType_ThinStructure	Defines the ThinStructure pattern type.
EPatternType_Unknown	-

EPickingMode Enum

-	-
---	---

EPickingMode_All	-
------------------	---

EPickingMode_Begin	-
EPickingMode_End	-
EPickingMode_Central	-
EPickingMode_Score	-

EPlotItem Enum

Defines how the profile is drawn across a gauge.

EPlotItem_Transitions	Displays the profile along a point location gauge.
EPlotItem_Peak	Displays the corresponding derivative curve.
EPlotItem_Thresholds	Displays the threshold and minimum amplitude levels.
EPlotItem_Points	Displays the valid transitions.

EQRCodeCodingMode Enum

This enumeration contains the possible values for the coding mode of a QR code.

EQRCodeCodingMode_Basic	The QR code does not use a specific coding mode.
EQRCodeCodingMode_Fnc1_Gs1	The QR code uses the FNC1/GS1 coding mode (FNC1 in first position).

EQRCodeCodingMode_Fnc1_Aim	The QR code uses the FNC1/AIM coding mode (FNC1 in second position).
----------------------------	--

EQRCodeEncoding Enum

This enumeration contains the possible values for the encoding used for parts of the bit stream of a QR code.

EQRCodeEncoding_Numeric	The stream part is coded numerically.
EQRCodeEncoding_Alphanumeric	The stream part is coded alphanumerically.
EQRCodeEncoding_Byt	The stream part is coded as bytes.
EQRCodeEncoding_Kanji	The stream part is coded in Kanji.

EQRCodeLevel Enum

This enumeration contains the possible values for the level of error correction of a QR code.

EQRCodeLevel_L	The QR code is level L (about 7% of error correction).
EQRCodeLevel_M	The QR code is level M (about 15% of error correction).
EQRCodeLevel_Q	The QR code is level Q (about 25% of error correction).

EQRCodeLevel_H

The QR code is level H (about 30% of error correction).

E

Q

RCodeModel Enum

This enumeration contains the possible values for a QR code model.

EQRCodeModel_Model1	The QR code is a model 1.
EQRCodeModel_Model2	The QR code is a model 2 or 2005.
EQRCodeModel_MicroQR	The QR code is a Micro QR.

EQRCodePerspectiveMode Enum

This enumeration contains the possible values for the scanning precision of a QR Code reader object.

EQRCodePerspectiveMode_Basic	The QR Code reader handles light perspective deformations.
EQRCodePerspectiveMode_Improved	The QR Code reader handles heavy perspective deformations.

Remarks

This setting has deprecated as per Open eVision release 2.0, setting the perspectiveMode will have no effect. The EQRCodePerspectiveMode_Basic option has been superceded by EQRDetectionMethod_GradientLegacy, the EQRCodePerspectiveMode_Improved option has been superceded by EQRDetectionMethod_PerspectiveLegacy.

EQRCodeScanPrecision Enum

This enumeration contains the possible values for the scanning precision of a QR code reader object.

EQRCodeScanPrecision_Automatic	The QR code reader determines the scan precision automatically.
EQRCodeScanPrecision_Fine	The QR code reader scans finely the search field to find the QR codes. This value is recommended for small images or large images with small QR codes.
EQRCodeScanPrecision_Coarse	The QR code reader scans coarsely the search field to find the QR codes. This value is recommended for large images with medium to large QR codes.

EQRDetectionMethod Enum

This enumeration contains the possible detection methods for QR codes, combinations of the methods are allowed.

EQRDetectionMethod_AdaptiveThreshold	This method detects finder patterns based on adaptive thresholding of the image.
EQRDetectionMethod_Gradient	This method detects finder patterns based on gradients in the image.

EQRDetectionMethod_PerspectiveLegacy	This selects the gradient-based detection algorithms with improved perspective mode developed for eVision 1.2.2.
EQRDetectionMethod_GradientLegacy	This selects the gradient-based detection algorithms with basic perspective mode developed for eVision 1.2.2.

Remarks

The variables Topology, CharHeight, CharWidth should be set before performing this operation.

EQRDetectionTradeOff Enum

This enumeration contains several settings for the tradeoff between detection speed and reliability of the easyQRCode methods. Setting this parameter will overwrite the current settings for EQRDetectionMethod and EQRCODEScanPrecision.

EQRDetectionTradeOff_FavorSpeed	This setting gives the fastest detection speed, but may reduce the detection accuracy. Sets EQRDetectionMethod_AdaptiveThreshold and EQRCODEScanPrecision_Coarse.
EQRDetectionTradeOff_Balanced	This setting gives the a balance between detection speed and reliability. Sets EQRDetectionMethod_AdaptiveThreshold EQRDetectionMethod_Gradient and EQRCODEScanPrecision_Automatic.
EQRDetectionTradeOff_FavorReliability	This setting gives the best detection reliability, at the cost of detection speed. Sets EQRDetectionMethod_AdaptiveThreshold

EQRDetectionTradeOff_Custom	EQRDetectionMethod_Gradient and EQRCODEScanPrecision_Fine.
This setting is returned when the current settings EQRDetectionMethod and EQRCODEScanPrecision do not match any of the EQRDetectionTradeOff presets. This choice should NOT be used to set a desired trade-off setting.	

EQualityIndicator Enum

Allowed values for the quality indicators in EasyOCV.

EQualityIndicator_Location	Global scores based on the edge pixels of the characters.
EQualityIndicator_Area	Foreground and background pixel counts.
EQualityIndicator_Sum	Sum of the normalized foreground and background gray-level values.
EQualityIndicator_Correlation	Normalized correlation.
EQualityIndicator_Contrast	Global contrast of the inspected ROI.

ERectangleMode Enum

The modes that specify how the selection of coded elements with a rectangle behaves.

ERectangleMode_EntirelyInside	Takes into consideration only the coded elements that entirely lie inside the given rectangle, not touching its borders
ERectangleMode_EntirelyOutside	Takes into consideration only the coded elements that entirely lie outside the given rectangle, not touching its borders.
ERectangleMode_InsideOrOnBorder	Takes into consideration only the coded elements that entirely lie inside the given rectangle, or that touch its borders.
ERectangleMode_OutsideOrOnBorder	Takes into consideration only the coded elements that entirely lie outside the given rectangle, or that touch its borders.
ERectangleMode_OnBorder	Takes into consideration only the coded elements that touch the borders of the rectangle.

EReductionMode Enum

The reduction mode to be used when learning a Consistent Edges model.

EReductionMode_Auto	Use the best-guess algorithm for selecting the reduction strength when learning a model.
EReductionMode_Manual	Use a user-set value for the reduction strength when learning a model (cf. the property EPatternFinder::ReductionStrength).
EReductionMode_Unknown	-

EReferenceNoise Enum

Enumeration for specifying how a reference image is affected by noise in EasyImage.

EReferenceNoise_NoReference	The reference image is free from noise (synthetic image or noise source cancelled).
EReferenceNoise_SameAsImage	The reference image is contaminated by the same noise source as the source image.

ERgbStandard Enum

Allowed values for the RGB standard in EasyColor.

ERgbStandard_Ntsc	NTSC primaries with the following CIE XYZ coordinates: Red: (0.607, 0.299, 0.000), Green: (0.174, 0.587, 0.066), Blue: (0.201, 0.114, 1.117).
ERgbStandard_Pal	PAL primaries with the following CIE XYZ coordinates: Red: (0.4303, 0.2219, 0.0202), Green: (0.3416, 0.7068, 0.1296), Blue: (0.1784, 0.0713, 0.9393).
ERgbStandard_Smpte	SMPTE primaries with the following CIE XYZ coordinates: Red: (0.393, 0.212, 0.019), Green: (0.365, 0.701, 0.112), Blue: (0.192, 0.087, 0.958).

Remarks

The definition of the RGB primaries is not unique. In principle, there is one RGB system for each set of phosphors used in color monitors. Anyway, the CCIR has defined standard combinations for use in digital TV broadcast. Before performing a conversion, function [EasyColor::RgbStandard](#) can be used to specify the standard used.

ERoiHit Enum

Describes the ROI that was hit by the mouse cursor.

ERoiHit_NoHit	No ROI.
ERoiHit_Learn_0	First learning ROI.
ERoiHit_Learn_1	Second learning ROI.
ERoiHit_Match_0	First matching ROI.
ERoiHit_Match_1	Second matching ROI.
ERoiHit_Inspect	Inspection ROI.

ESegmentationMethod Enum

The segmentation methods that are available to the image encoder.

ESegmentationMethod_Custom	-
ESegmentationMethod_BinaryImage	Segmentation of binary images (cf. EBinaryImageSegmenter).
ESegmentationMethod_ColorRangeThreshold	Segments a color image by specifying a cube in the RGB space (cf. EColorRangeThresholdSegmenter).

E SegmentationMethod_ColorSingleThreshold	Segments a color image by specifying a single threshold (cf. EColorSingleThresholdSegmenter).
E SegmentationMethod_GrayscaleDoubleThreshold	Segments a grayscale image by specifying a double threshold (cf. EGayscaleDoubleThresholdSegmenter).
E SegmentationMethod_GrayscaleSingleThreshold	Segments a grayscale image by specifying a single threshold (cf. EGayscaleSingleThresholdSegmenter).
E SegmentationMethod_ImageRange	Segments an image by specifying a pixel-by-pixel double threshold (cf. EGayscaleDoubleThresholdSegmenter).
E SegmentationMethod_ReferenceImage	Segments an image by specifying a pixel-by-pixel single threshold (cf. EGayscaleSingleThresholdSegmenter).
E SegmentationMethod_LabeledImage	Segments an image by mapping the value of the pixels directly to a layer index (cf. ELabeledImageSegmenter).

Remarks

The parameters of the segmentation methods are configured through the getters finishing by "Segmenter" that are available in [EImageEncoder](#).

E SegmentationMode Enum

Allowed values for the segmentation mode in EasyOCR.

E SegmentationMode_KeepObjects

E SegmentationMode_	After segmentation, keep the blobs as they were found.
RepasteObjects	After segmentation, group together the blobs believed to belong to the same character.

ESelectByPosition Enum

Allowed values for the selection mode of [ECodedImage](#).

ESelectByPosition_InsertIn	Insert the objects completely inside the given area.
ESelectByPosition_InsertTouch	Insert all the objects with a non-empty intersection with the given area.
ESelectByPosition_InsertOut	Insert the objects completely outside the given area.
ESelectByPosition_RemoveIn	Remove the objects completely inside the given area.
ESelectByPosition_RemoveTouch	Remove all the objects with a non-empty intersection with the given area.
ESelectByPosition_RemoveOut	Remove the objects completely outside the given area.
ESelectByPosition_RemoveBorder	Remove the objects outside the given area (including the objects touching the given area boundary).

Remarks

When specifying the position by means of an ROI, the minimum width and height of the ROI object must be at least 3 pixels. This enumeration pertains to the EasyObject legacy API and should not be used for new developments. See [ECodedImage2](#) for the new API.

ESelectionFlag Enum

Specifies to which subset of a selection an operation should be applied in EasyObject and EasyOCV.

ESelectionFlag_Any	The operation applies to both selected and unselected items.
ESelectionFlag_True	The operation applies to selected items only.
ESelectionFlag_False	The operation applies to unselected items only.

ESelectOption Enum

Allowed values for the selection mode of [ECodedImage](#). This enumeration pertains to the EasyObject legacy API and should not be used for new developments. See [ECodedImage2](#) for the new API.

ESelectOption_InsertAll	Add all objects.
ESelectOption_InsertGreaterOrEqual	Add all objects with feature value above the upper threshold.

ESelectOption_InsertLesserOrEqual	Add all objects with feature value below the lower threshold.
ESelectOption_InsertRange	Add all objects with feature value between or equal to the lower and upper thresholds.
ESelectOption_RemoveAll	Delete all objects.
ESelectOption_RemoveGreaterOrEqual	Delete all objects with feature value above the lower threshold.
ESelectOption_RemoveLesserOrEqual	Delete all objects with feature value below the upper threshold.
ESelectOption_RemoveRange	Delete all objects with feature value between or equal to the lower and upper thresholds.
ESelectOption_InsertOutOfRange	Add all objects with feature value above the upper and below the lower threshold.
ESelectOption_RemoveOutOfRange	Delete all objects with feature value above the upper and below the lower threshold.

ESerializer.FileWriterMode Enum

Creation mode of the file.

ESerializer.FileWriterMode_Create	Creates the archive file on the hard disk. If the file already exists, the ESerializer::CreateFileWriter method returns NULL .

ESerializer.FileWriterMode_Overwrite	Overwrites the previously created archive if it already exists, or creates it otherwise.
ESerializer.FileWriterMode_Append	Appends the data at the end of the previously created archive, or creates the archive if it doesn't exist.

EShapeBehavior Enum

Allowed values for conditions on the behavior of a shape.

EShapeBehavior_Visible	Identifies a visible shape.
EShapeBehavior_Selected	Identifies a selected shape.
EShapeBehavior_Selectable	Identifies a selectable shape.
EShapeBehavior_Dragable	Identifies a dragable shape.
EShapeBehavior_Rotatable	Identifies a rotatable shape.
EShapeBehavior_Resizable	Identifies a resizable shape.
EShapeBehavior_Labeled	Identifies a labeled shape.
EShapeBehavior_Active	Identifies an active shape.
EShapeBehavior_Passed	Identifies a non defective shape.

EShapeType Enum

Gauge type.

EShapeType_NoShape	-
EShapeType_FrameShape	Defines a frame shape.
EShapeType_WorldShape	-
EShapeType_PointGauge	Defines a point location gauge.
EShapeType_LineGauge	Defines a line fitting gauge.
EShapeType_CircleGauge	Defines a circle fitting gauge.
EShapeType_RectangleGauge	Defines a rectangle fitting gauge.
EShapeType_WedgeGauge	Defines a wedge fitting gauge.

EShiftingMode Enum

Allowed values for the shifting mode of EasyOCR.

EShiftingMode_Chars	Each character is moved individually.
EShiftingMode_Text	The all set of characters is moved together.

ESingleThresholdMode Enum

The single threshold mode for the selection of coded elements with respect to a given feature.

ESingleThresholdMode_Less	The value of the feature must be strictly less than the threshold.
ESingleThresholdMode_LessEqual	The value of the feature must be less or equal to the threshold.
ESingleThresholdMode_Equal	The value of the feature must be equal to the threshold.
ESingleThresholdMode_GreaterEqual	The value of the feature must be greater or equal to the threshold.
ESingleThresholdMode_Greater	The value of the feature must be strictly greater than the threshold.
ESingleThresholdMode_Different	The value of the feature must be different to the threshold.

ESortDirection Enum

The sorting mode for selections of coded elements based on their features.

ESortDirection_Ascending	Sorts the coded elements with respect to a given feature, in ascending order.
ESortDirection_Descending	

ESortOption Enum

Allowed values for the sort mode of [ECodedImage](#). This enumeration pertains to the EasyObject legacy API and should not be used for new developments. See [ECodedImage](#) for the new API.

ESortOption_Ascending	Sort by increasing feature values.
ESortOption_Descending	Sort by decreasing feature values.

EStockMeasurementUnit Enum

Allowed values for the type of Measurement Unit.

EStockMeasurementUnit_None	Defines the None value type.
EStockMeasurementUnit_um	Defines the micron meter value type.
EStockMeasurementUnit_mm	Defines the milimeter value type.
EStockMeasurementUnit_cm	Defines the centimeter value type.
EStockMeasurementUnit_dm	Defines the decimeter value type.
EStockMeasurementUnit_m	Defines the meter value type.
EStockMeasurementUnit_dam	Defines the decameter value type.

EStockMeasurementUnit_nm	Defines the Hectometer value type.
EStockMeasurementUnit_km	Defines the Kilometer value type.
EStockMeasurementUnit_mil	Defines the milliliter value type.
EStockMeasurementUnit_inch	Defines the inch value type.
EStockMeasurementUnit_foot	Defines the foot value type.
EStockMeasurementUnit_yard	Defines the yard value type.
EStockMeasurementUnit_mile	Defines the mile value type.

ESymbologies Enum

The symbologies supported by EasyBarcode.

ESymbologies_Standard_Symbologies	Reserved for internal use
ESymbologies_Additional_Symbologies	Reserved for internal use
ESymbologies_Codabar	Codabar symbology.
ESymbologies_Code128	Code 128 symbology.
ESymbologies_Code25Interleaved	Code 25 Interleaved symbology.
ESymbologies_Code39	Code 39 symbology.

ESymbologies_Ean128	EAN 128 symbology.
ESymbologies_Ean13	EAN 13 symbology.
ESymbologies_Msi	MSI symbology.
ESymbologies_UpcA	UPC A symbology.
ESymbologies_UpcE	UPC E symbology.
ESymbologies_BinaryCode	Binary Code symbology.
ESymbologies_AdsAnker	Code ABC Anker symbology.
ESymbologies_Bc412	Code BC 412 symbology.
ESymbologies_Code11	Code 11 symbology.
ESymbologies_Code13	Code 13 symbology.
ESymbologies_Code25Datalogic	Code 25 DataLogic symbology.
ESymbologies_Code25Matrix	Code 25 Matrix symbology.
ESymbologies_Code25Iata	Code 25 IATA symbology.
ESymbologies_Code25Industry	Code 25 Industry symbology.
ESymbologies_Code25Compressed	Code 25 Compressed symbology.
ESymbologies_Code25Inverted	Code 25 Inverted symbology.
ESymbologies_Code32	Code 32 symbology.

ESymbologies_Code39Extended	Code 39 Extended symbology.
ESymbologies_Code39Reduced	Code 39 Reduced symbology.
ESymbologies_Code93	Code 93 symbology.
ESymbologies_Code93Extended	Code 93 Extended symbology.
ESymbologies_CodeBcdMatrix	Code BCD Matrix symbology.
ESymbologies_CodeCip	Code CIP symbology.
ESymbologies_CodeStk	Code STK symbology.
ESymbologies_Ean8	EAN 8 symbology.
ESymbologies_IbmDeltaDistanceA	IBM Delta Distance A symbology.
ESymbologies_Plessey	Plessey symbology.
ESymbologies_Telepen	Telepen symbology.
ESymbologies_Rss14	RSS-14 symbology.
ESymbologies_Rss14Limited	RSS-14 Limited symbology.
ESymbologies_Rss14Expanded	RSS-14 Expanded symbology.
ESymbologies_Standard	Gathers all the symbologies belonging to the standard group.
ESymbologies_Additional	Gathers all the symbologies belonging to the additional group.
ESymbologies_Unknown	-

Remarks

Due to the large number of supported symbologies, they have been splitted into two groups. The most commonly used symbologies have been gathered under the name Standard symbologies. The remaining symbologies belong to the Additional symbologies group.

EThinStructureMode Enum

Allowed values for the type of thin structures in EasyFind.

EThinStructureMode_Auto	Lets EasyFind choose automatically the best contrast of thin elements.
EThinStructureMode_Dark	Favors thin elements darker than regions.
EThinStructureMode_Bright	Favors thin elements brighter than regions.

EThresholdMode Enum

The various modes for thresholding that are supported by Open eVision.

EThresholdMode_Absolute	Reserved value. For absolute thresholding, use the threshold value itself, cast to EThresholdMode .
EThresholdMode_Relative	Relative threshold; determines the required threshold level so that a given fraction of the image pixels lie below it.
EThresholdMode_MinResidue	Selects a threshold value such that the quadratic difference between the source and thresholded image is minimized.

EThresholdMode_MaxEntropy	Selects a threshold value such that the entropy (i.e. the amount of information) of the resulting thresholded image is maximized.
EThresholdMode_Isodata	Selects a threshold value that lies halfway between the average dark gray value (i.e. gray levels below the threshold) and average light gray values (i.e. gray levels above the threshold).

ETransitionChoice Enum

The transition selection method applied by the gauge measurement

ETransitionChoice_NthFromBegin	N-th transition from the beginning (counting from 0).
ETransitionChoice_NthFromEnd	N-th transition from the end (counting from 0).
ETransitionChoice_LargestAmplitude	Transition whose peak has the largest amplitude value.
ETransitionChoice_LargestArea	Transition whose peak has the largest area value.
ETransitionChoice_Closest	Transition closest to the center.
ETransitionChoice_All	All transitions.

ETransitionType Enum

The type of transition to be retained by the gauge measurement

ETransitionType_Bw	Black to white.
ETransitionType_Wb	White to black.
ETransitionType_BwOrWb	Black to white or white to black.
ETransitionType_Bwb	Black to white to black.
ETransitionType_Wbw	White to black to white.

Features Enum

Open eVision Features

EasyGauge	-
EasyColor	-
EasylImage	-
EasyObject	-
EasyBarCode	-
EasyMatch	-
eVisionStudio	-
EasyFind	-
EasyMatrixCode	-

EasyOCR	-
EasyOCV	-
EasyQRCode	-
EasyOCR2	-