JavaScript

PhenoCapture Scripting API Reference — JavaScript (ClearScript)

Browse classes, categories, and methods below, or use the floating search bar to locate functions sequentially. 9 Classes • 494 Functions

1. Action (ScriptActionHelper) 130 Functions

Color & Channel Actions 25 functions

1. Action.AdaptiveContrast(contrastStrengthPercentage, gridTilesX, gridTilesY, contrastClipLimit)

Returns: Boolean

DESCRIPTION

Enhances local contrast using CLAHE.

SYNTAX

```
let result = Action.AdaptiveContrast(contrastStrengthPercentage, gridTilesX, gridTilesY, contrastClipLimit);
```

PARAMETERS

Parameter	Description	Default
`contrastStrengthPercentage` Integer, Optional	Contrast enhancement strength percentage.	20
`gridTilesX` Integer, Optional	Number of grid tiles horizontally.	8
`gridTilesY` Integer, Optional	Number of grid tiles vertically.	8
`contrastClipLimit` Single, Optional	CLAHE histogram clip limit.	1.0F

RETURN VALUE

Boolean — True if contrast enhancement succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.AdaptiveContrast(20, 8, 8, 1.0F);
```

2. Action.AutoContrast(clipPercentage) Returns: Boolean

DESCRIPTION

Automatically adjusts image contrast.

SYNTAX

```
let result = Action.AutoContrast(clipPercentage);
```

PARAMETERS

Parameter	Description	Default
clipPercentage Single, Optional	Histogram clipping percentage.	0.5F

RETURN VALUE

Boolean — True if contrast adjustment succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.AutoContrast(0.5F);
```

3. Action.AutoEqualize() Returns: Boolean

DESCRIPTION

Equalizes the image histogram across the tonal range.

SYNTAX

```
let result = Action.AutoEqualize();
```

RETURN VALUE

Boolean — True if equalization succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.AutoEqualize();
```

4. Action.AutoGamma() Returns: Boolean

DESCRIPTION

Automatically adjusts image gamma.

SYNTAX

Search

Search

Latest Posts

AI Face Detection

AI Motion Deblur

Privacy Policy

Terms of Service

Hangul File Preview & Import (HWP, HWPX)

Split Toning

Highlight Bloom & Halation

Miniature Effect

High-Pass Sharpening

Dehaze

Smart Sketch

Orton Effect

HDR Local Tone Mapping

Soft Glow

ESP32-CAM Remote Acquisition Tool

Stack Blur

Color Deconvolution

Selection Tool

AI Exposure Correction

Remove Dots

AI Object Detection

Crop and Export Regions

Paste Selection As Binary

Quick Start: Workspace Setup and Basic Use

Image Quality Restoration

Paste and Fit to Rectangle Selection

Gamma Adjustment

Inverse Fourier Transform (IFFT)

Paste

Copy

Image Stamp

Brush Annotation

Toolbox

Unselect

Undo & Redo Functions

Subtract Binary Image from Selection

Add Binary Image to Selection

Create Selection From Binary Image

Shrink Selection

Expand Selection

Reverse Selection

Magnetic Snap & Auto-Routing

AI Background Removal (General Object)

AI Background Removal (Portrait)

AI Background Removal (Selfie)

AI Auto Subject Selection (General Object)

AI Auto Subject Selection (Portrait)

AI Smart Selection

Custom AI Binarization Trainer

Custom AI Binarization

AI Binarization and Segmentation

AI OCR Text Extraction

AI Document Unwarp

AI Inpainting

AI Depth Estimation

Change Non-black Pixels to White

Edge Detection – Sobel, Left-hand, Upper-Left-hand

Edge Detection – Homogenity

Canny Edge Detection

Fast Fourier Transform (Magnitude)

© 2026 **PhenoCapture AI**. All rights reserved.
Terms of Service | Privacy Policy | Contact Us

```
let success = Action.ChangeColor(255, 255, 255, 0, 0, 0);
```

10. Action.CIELab(lightnessShift, aAxisShift, bAxisShift)

Returns: Boolean

DESCRIPTION

Adjusts image color in the perceptually uniform CIE Lab space.

SYNTAX

```
let result = Action.CIELab(lightnessShift, aAxisShift, bAxisShift);
```

PARAMETERS

Parameter	Description	Default
`lightnessShift` Integer, Optional	Change to the L lightness axis.	0
`aAxisShift` Integer, Optional	Change to the a color axis.	0
`bAxisShift` Integer, Optional	Change to the b color axis.	0

RETURN VALUE

Boolean — True if the adjustment was applied; otherwise False.

EXAMPLE CODE

EXAMPLE CODE

```
let success = Action.CIELab(0, 0, 0);
```

11. Action.ColorDeconvolution(stainType, showColor)

Returns: Boolean

DESCRIPTION

Applies color deconvolution to separate stains in the active workspace image.

SYNTAX

```
let result = Action.ColorDeconvolution(stainType, showColor);
```

PARAMETERS

Parameter	Description	Default
`stainType` Integer	Integer representing the biological stain to extract. Supported stainType values: 0: H&E - Hematoxylin (Nuclei) 1: H&E - Eosin (Cytoplasm) 2: H-DAB - Hematoxylin (Background) 3: H-DAB - DAB (Protein Marker) 4: Masson's Trichrome - Methyl Blue (Collagen) 5: Masson's Trichrome - Ponceau Fuchsin (Muscle) 6: H-AEC - Hematoxylin (Nuclei) 7: H-AEC - AEC (Red Marker) 8: Methyl Green-DAB - Methyl Green (Nuclei) 9: Methyl Green-DAB - DAB (Protein Marker) 10: H-PAS - Hematoxylin (Nuclei) 11: H-PAS - PAS (Glycogen / Basement Membrane) 12: Alcian Blue & H - Alcian Blue (Acid Mucins / GAG) 13: Alcian Blue & H - Hematoxylin (Nuclei) 14: Giemsa - Methylene Blue (Nuclei) 15: Giemsa - Eosin (Erythrocytes / Cytoplasm) 16: Feulgen-Light Green - Feulgen (DNA) 17: Feulgen-Light Green - Light Green (Cytoplasm)	–

	(Cytoplasm) 18: Safranin-O & Fast Green - Safranin-O (Lignin / Cartilage) 19: Safranin-O & Fast Green - Fast Green (Cellulose / Cytoplasm) 20: H&E-DAB (Triple) - Hematoxylin (Nuclei) 21: H&E-DAB (Triple) - Eosin (Cytoplasm) 22: H&E-DAB (Triple) - DAB (Protein Marker)	
showColor Boolean	Whether to apply pseudo-coloring to the extracted stain result.	–

RETURN VALUE

Boolean — True if the deconvolution process succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.ColorDeconvolution(0, true);
```

12. Action.ColorRangeSelection(targetColorRed, targetColorGreen, targetColorBlue, colorSpaceType, algorithmType, tolerance, keepOriginalColor)

Returns: Boolean

DESCRIPTION

Selects or isolates image pixels matching the specified target RGB color within a given tolerance.

SYNTAX

```
let result = Action.ColorRangeSelection(targetColorRed, targetColorGreen, targetColorBlue, colorSpaceType, algorithmType, tolerance, keepOriginalColor);
```

PARAMETERS

Parameter	Description	Default
targetColorRed Integer, Optional	Target color red component (0–255).	0
targetColorGreen Integer, Optional	Target color green component (0–255).	255
targetColorBlue Integer, Optional	Target color blue component (0–255).	0

`colorSpaceType` Integer, Optional	Integer representing the color space model. Supported colorSpaceType values: 0: RGB 1: HSI 2: Lab 3: Chroma	0
`algorithmType` Integer, Optional	Integer representing the distance calculation algorithm. Supported algorithmType values: 0: Fast - Manhattan distance 1: Standard - Euclidean distance (Ignored when colorSpaceType is 3: Chroma)	1
`tolerance` Integer, Optional	Color matching tolerance range (0–255).	40
`keepOriginalColor` Boolean, Optional	Whether to preserve original pixel colors for matched areas (False converts matched areas to white).	True

RETURN VALUE

Boolean — True if the color range selection was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.ColorRangeSelection(0, 255, 0, 3, 1, 40, true);
```

13. Action.ConvertToRgChrom() Returns: Boolean

DESCRIPTION

Converts the image to RG chromaticity coordinates.

SYNTAX

```
let result = Action.ConvertToRgChrom();
```

RETURN VALUE

Boolean — True if conversion succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.ConvertToRgChrom();
```

14. Action.CorrectUnevenIllumination(illuminationGridSize)

Returns: Boolean

DESCRIPTION

Corrects uneven image illumination using a rolling-ball algorithm.

SYNTAX

```
let result = Action.CorrectUnevenIllumination(illuminationGridSize);
```

PARAMETERS

Parameter	Description	Default
`illuminationGridSize` Integer, Optional	Grid size used for illumination estimation.	8

RETURN VALUE

Boolean — True if illumination correction succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.CorrectUnevenIllumination(8);
```

15. Action.DeleteChannel(channelToDeleteType) Returns: Boolean

DESCRIPTION

Removes the specified color channel from the image.

SYNTAX

```
let result = Action.DeleteChannel(channelToDeleteType);
```

PARAMETERS

Parameter	Description	Default
`channelToDeleteType` Integer, Optional	Integer representing the color channel. Supported channelToDeleteType values: 0: Red 1: Green 2: Blue	0

RETURN VALUE

Boolean — True if the channel was removed; otherwise False.

EXAMPLE CODE

```
let success = Action.DeleteChannel(0);
```

16. Action.GrayFrom(sourceChannelType) Returns: Boolean

DESCRIPTION

Creates a grayscale image from the selected color channel.

SYNTAX

```
let result = Action.GrayFrom(sourceChannelType);
```

PARAMETERS

Parameter	Description	Default
sourceChannelType Integer, Optional	Integer representing the source color channel. Supported sourceChannelType values: 0: Red 1: Green 2: Blue	0

RETURN VALUE

Boolean — True if conversion succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.GrayFrom(0);
```

17. Action.GrayScaling() Returns: Boolean

DESCRIPTION

Converts the image to grayscale.

SYNTAX

```
let result = Action.GrayScaling();
```

RETURN VALUE

Boolean — True if conversion succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.GrayScaling();
```

18. Action.GrayToColor(mapType, isReversed) Returns: Boolean

DESCRIPTION

Applies a color map to a grayscale image.

SYNTAX

```
let result = Action.GrayToColor(mapType, isReversed);
```

PARAMETERS

Parameter	Description	Default
mapType Integer, Optional	Integer representing the color map. Supported mapType values: 0: RedBlue 1: Viridis 2: Parula 3: Plasma 4: Inferno 5: Magma 6: Cividis 7: Jet 8: Turbo 9: Rainbow 10: Spectral 11: HSV 12: Ocean 13: Seismic 14: RdBu 15: Coolwarm 16: bwr 17: BrBG 18: PiYG 19: Hot 20: Cool 21: Spring 22: Summer 23: Autumn 24: Winter 25: Bone 26: Copper 27: Pink	0

	28: Terrain 29: Twilight 30: Rocket (Black - Blue - Red - Light Yellow) 31: Mako (Black - Deep Blue - Purple - Light Pink) 32: Cubehelix (Astronomy standard, brightness preserved in grayscale) 33: YlGnBu (Yellow - Green - Blue) 34: RdYlGn (Red - Yellow - Green) 35: PuOr (Purple - White - Orange) 36: Ironbow (Thermal: Black - Purple - Red - Yellow - White) 37: WhiteHot (Thermal: Cold Black - Hot White) 38: BlackHot (Thermal: Cold White - Hot Black) 39: Lava (Black - Dark Red - Orange - Yellow - White) 40: Arctic (Black - Dark Blue - Cyan - White)	
isReversed Boolean, Optional	Whether to reverse the color map.	False

RETURN VALUE

Boolean — True if the color map was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.GrayToColor(0, false);
```

19. Action.HueSaturationIntensity(hue, saturation, intensity)

Returns: Boolean

DESCRIPTION

Adjusts hue, saturation, and intensity independently.

SYNTAX

```
let result = Action.HueSaturationIntensity(hue, saturation, intensity);
```

PARAMETERS

Parameter	Description	Default

hue Integer, Optional	Hue adjustment.	0
saturation Integer, Optional	Saturation adjustment.	0
intensity Integer, Optional	Intensity adjustment.	0

RETURN VALUE

Boolean — True if the adjustment was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.HueSaturationIntensity(0, 0, 0);
```

20. Action.Invert() Returns: Boolean

DESCRIPTION

Inverts all image colors to create a negative.

SYNTAX

```
let result = Action.Invert();
```

RETURN VALUE

Boolean — True if colors were inverted; otherwise False.

EXAMPLE CODE

```
let success = Action.Invert();
```

21. Action.MergeChannels(colorSpaceType, image1Source, image2Source, image3Source)

Returns: Boolean

DESCRIPTION

Merges three image sources as channels in the selected color space.

SYNTAX

```
let result = Action.MergeChannels(colorSpaceType, image1Source, image2Source, image3Source);
```

PARAMETERS

Parameter	Description	Default
`colorSpaceType` Integer, Optional	Integer representing the target color space. Supported colorSpaceType values: 0: RGB 1: HSI 2: LAB	0
`image1Source` Integer, Optional	First source image index.	0
`image2Source` Integer, Optional	Second source image index.	1
`image3Source` Integer, Optional	Third source image index.	2

RETURN VALUE

Boolean — True if the channels were merged; otherwise False.

EXAMPLE CODE

```
let success = Action.MergeChannels(0, 0, 1, 2);
```

22. Action.MultibandColor(rSat, rInt, ySat, yInt, gSat, gInt, cyanSat, cyanInt, bSat, bInt, mSat, mInt)

Returns: Boolean

DESCRIPTION

Adjusts saturation and intensity for red, yellow, green, cyan, blue, and magenta bands.

SYNTAX

```
let result = Action.MultibandColor(rSat, rInt, ySat, yInt, gSat, gInt, cyanSat, cyanInt, bSat, bInt, mSat, mInt);
```

PARAMETERS

Parameter	Description	Default
`rSat` Integer, Optional	Red-band saturation adjustment.	0
`rInt` Integer, Optional	Red-band intensity adjustment.	0

ySat Integer, Optional	Yellow-band saturation adjustment.	0
yInt Integer, Optional	Yellow-band intensity adjustment.	0
gSat Integer, Optional	Green-band saturation adjustment.	0
gInt Integer, Optional	Green-band intensity adjustment.	0
cyanSat Integer, Optional	Cyan-band saturation adjustment.	0
cyanInt Integer, Optional	Cyan-band intensity adjustment.	0
bSat Integer, Optional	Blue-band saturation adjustment.	0
bInt Integer, Optional	Blue-band intensity adjustment.	0
mSat Integer, Optional	Magenta-band saturation adjustment.	0
mInt Integer, Optional	Magenta-band intensity adjustment.	0

RETURN VALUE

Boolean — True if the adjustment was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.MultibandColor(0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0);
```

23. Action.RollingBallBackgroundSubtraction(rollingBallBoxSize, isObjectColorAutomatic, manualObjectColorType)

Returns: Boolean

DESCRIPTION

Removes background using rolling-ball subtraction.

SYNTAX

```
let result = Action.RollingBallBackgroundSubtraction(rollingBallBoxSize, isObjectColorAutomatic, manualObjectColorType);
```

PARAMETERS

Parameter	Description	Default
`rollingBallBoxSize` Integer, Optional	Rolling-ball box size in pixels.	50
`isObjectColorAutomatic` Boolean, Optional	Whether to detect object color automatically.	True
`manualObjectColorType` Integer, Optional	Integer representing the manual object color. Supported manualObjectColorType values (used when automatic detection is disabled): 0: White 1: Black	0

RETURN VALUE

Boolean — True if background subtraction succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.RollingBallBackgroundSubtraction(50, true, 0);
```

24. Action.SplitRGB(rgbSplitChannelType) Returns: Boolean

DESCRIPTION

Splits the image into the selected RGB channel.

SYNTAX

```
let result = Action.SplitRGB(rgbSplitChannelType);
```

PARAMETERS

Parameter	Description	Default
`rgbSplitChannelType` Integer, Optional	Integer representing the target channel. Supported rgbSplitChannelType values: 0: RGB Red	0

1: RGB Green
2: RGB Blue
3: Hue
4: Saturation
5: Intensity
6: CMYK Cyan
7: CMYK Magenta
8: CMYK Yellow
9: CMYK Key (Black)
10: Lab L
11: Lab a
12: Lab b
13: rg chromaticity r
14: rg chromaticity g
15: rg chromaticity b

RETURN VALUE

Boolean — True if the channel was extracted; otherwise False.

EXAMPLE CODE

```
let success = Action.SplitRGB(0);
```

25. Action.SplitToning(shadowHue, highlightHue, toneStrength, tonalBalance)

Returns: Boolean

DESCRIPTION

Applies dual-tone color grading to shadows and highlights independently using a high-speed 256-entry luminance LUT.

SYNTAX

```
let result = Action.SplitToning(shadowHue, highlightHue, toneStrength, tonalBalance);
```

PARAMETERS

Parameter	Description	Default
shadowHue Integer, Optional	Color hue angle in degrees for shadows. Range: 0 to 360.	210
highlightHue Integer, Optional	Color hue angle in degrees for highlights. Range: 0 to 360.	35
toneStrength Single, Optional	Overall color toning intensity. Range: 0.0 to 1.0	0.4F

Single, Optional	to 1.0.	
tonalBalance Integer, Optional	Balance shift between shadow and highlight tints. Range: -100 to 100.	0

RETURN VALUE

Boolean — True if the Split Toning filter was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.SplitToning(210, 35, 0.4F, 0);
```

Data Extraction & Analysis Actions 5 functions

1. Action.AverageColorValue(analysisTargetChannelType)

Returns: String

DESCRIPTION

Extracts the average value of the selected color channel.

SYNTAX

```
let result = Action.AverageColorValue(analysisTargetChannelType);
```

PARAMETERS

Parameter	Description	Default
analysisTargetChannelType Integer, Optional	Integer representing the target channel. Supported analysisTargetChannelType values: 0: RGB Red 1: RGB Green 2: RGB Blue 3: Hue 4: Saturation 5: Intensity 6: CMYK Cyan 7: CMYK Magenta 8: CMYK Yellow 9: CMYK Key (Black) 10: Lab L 11: Lab a 12: Lab b	0

13: rg chromaticity r
14: rg chromaticity g
15: rg chromaticity b

RETURN VALUE

`String` — String containing the extracted value, or an empty string if extraction fails.

EXAMPLE CODE

```
let average = Action.AverageColorValue(0);
```

2. Action.CropExportRegions(maskSourceLocation, connectivityType, marginPixels, outputModeType, transparentBackground, backgroundColorRed, backgroundColorGreen, backgroundColorBlue, destinationFolder, filePrefix, imageFormatType)

Returns: Boolean

DESCRIPTION

Exports individual regions identified by a binary mask image.

SYNTAX

```
let result = Action.CropExportRegions(maskSourceLocation, connectivityType, marginPixels, outputModeType, transparentBackground, backgroundColorRed, backgroundColorGreen, backgroundColorBlue, destinationFolder, filePrefix, imageFormatType);
```

PARAMETERS

Parameter	Description	Default
maskSourceLocation Integer, Optional	Mask image source buffer index.	1
connectivityType Integer, Optional	Integer representing the connected-component connectivity. Supported connectivityType values: 0: FourWay (4-connected) 1: EightWay (8-	0

	connected)	
marginPixels Integer, Optional	Extra margin around each exported region.	0
outputModeType Integer, Optional	Integer representing the region export output mode. Supported outputModeType values: 0: ROI - rectangular region of interest 1: ExactShape - exact region shape 2: Binary - binary mask	1
transparentBackground Boolean, Optional	Whether exported backgrounds are transparent.	True
backgroundColorRed Integer, Optional	Background red component (0–255).	0
backgroundColorGreen Integer, Optional	Background green component (0–255).	0
backgroundColorBlue Integer, Optional	Background blue component (0–255).	0
destinationFolder String, Optional	Directory for exported region files.	"C:₩₩ExportedRegions"
filePrefix String, Optional	Prefix for exported file names.	"RegionW_"
imageFormatType Integer, Optional	Integer representing the export image format. Supported imageFormatType values: 0: PNG 1: JPEG	0

2: BMP

RETURN VALUE

Boolean — True if region export succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.CropExportRegions(1, 0, 0, 1, true, 0, 0, 0,
"C:₩₩ExportedRegions", "Region₩_", 0);
```

3. Action.MultiCellAnalysis(gridColumnsX, gridRowsY, cellShapeType, cellDiameterPixels, analysisModeType, analysisTargetChannelType, targetColorPresetName)

Returns: String

DESCRIPTION

Analyzes cells in a grid and extracts per-cell statistics.

SYNTAX

```
let result = Action.MultiCellAnalysis(gridColumnsX, gridRowsY, ce
llShapeType, cellDiameterPixels, analysisModeType, analysisTarget
ChannelType, targetColorPresetName);
```

PARAMETERS

Parameter	Description	Default
`gridColumnsX` Integer, Optional	Number of grid columns.	5
`gridRowsY` Integer, Optional	Number of grid rows.	5
`cellShapeType` Integer, Optional	Integer representing the cell shape. Supported cellShapeType values: 0: Rectangle 1: Circle 2: Point	1
`cellDiameterPixels` Integer, Optional	Cell diameter in pixels.	20
`analysisModeType` Integer, Optional	Integer representing the analysis mode. Supported	0

	Supported analysisModeType values: 0: AverageColorValue 1: PixelCount	
analysisTargetChannelType Integer, Optional	Integer representing the target channel. Supported analysisTargetChannelType values: 0: RGB Red 1: RGB Green 2: RGB Blue 3: Hue 4: Saturation 5: Intensity 6: CMYK Cyan 7: CMYK Magenta 8: CMYK Yellow 9: CMYK Key (Black) 10: Lab L 11: Lab a 12: Lab b 13: rg chromaticity r 14: rg chromaticity g 15: rg chromaticity b	0
targetColorPresetName String, Optional	Target color preset name.	"White"

RETURN VALUE

String — String containing analysis data, or an empty string if analysis fails.

EXAMPLE CODE

```
let data = Action.MultiCellAnalysis(5, 5, 1, 20, 0, 0, "White");
```

4. Action.OCRTextExtraction() Returns: String

DESCRIPTION

Extracts text from the workspace image using OCR.

SYNTAX

```
let result = Action.OCRTextExtraction();
```

RETURN VALUE

String — Extracted text, an empty string when no text is found, or an error

String — Extracted text, an empty string when no text is found, or an error message.

EXAMPLE CODE

```
let text = Action.OCRTextExtraction();
```

5. Action.PixelCounter(targetColorRed, targetColorGreen, targetColorBlue, resultOutputFormatType)

Returns: String

DESCRIPTION

Counts pixels matching the specified RGB color.

SYNTAX

```
let result = Action.PixelCounter(targetColorRed, targetColorGreen, targetColorBlue, resultOutputFormatType);
```

PARAMETERS

Parameter	Description	Default
targetColorRed Integer, Optional	Target color red component (0–255).	255
targetColorGreen Integer, Optional	Target color green component (0–255).	255
targetColorBlue Integer, Optional	Target color blue component (0–255).	255
resultOutputFormatType Integer, Optional	Integer representing the result output format. Supported resultOutputFormatType values: 0: TotalCount 1: Percentage	0

RETURN VALUE

String — String containing the count results, or an empty string if extraction fails.

EXAMPLE CODE

```
let counts = Action.PixelCounter(255, 255, 255, 0);
```

Edit & Selection Actions 13 functions

1. Action.AddSelectionFromBinaryImage(sourceImageLocation)

Returns: Boolean

DESCRIPTION

Adds a binary image mask to the current selection using logical OR.

SYNTAX

```
let result = Action.AddSelectionFromBinaryImage(sourceImageLocation);
```

PARAMETERS

Parameter	Description	Default
`sourceImageLocation` Integer, Optional	Source buffer index from 1 to 10.	1

RETURN VALUE

Boolean — True if the selection was updated; otherwise False.

EXAMPLE CODE

```
let success = Action.AddSelectionFromBinaryImage(1);
```

2. Action.Copy() Returns: Boolean

DESCRIPTION

Copies the current image to the clipboard.

SYNTAX

```
let result = Action.Copy();
```

RETURN VALUE

Boolean — True if the image was copied; otherwise False.

EXAMPLE CODE

```
let success = Action.Copy();
```

3. Action.CreateSelectionFromBinaryImage(sourceImageLocation)

Returns: Boolean

DESCRIPTION

Creates a selection mask from a binary workspace or buffer image.

SYNTAX

```
let result = Action.CreateSelectionFromBinaryImage(sourceImageLocation);
```

PARAMETERS

Parameter	Description	Default
`sourceImageLocation` Integer, Optional	0 selects the workspace; 1–10 selects a buffer.	0

RETURN VALUE

Boolean — True if the selection was created; otherwise False.

EXAMPLE CODE

```
let success = Action.CreateSelectionFromBinaryImage(1);
```

4. Action.ExpandSelection(expandByPixels) Returns: Boolean

DESCRIPTION

Expands the current selection by a specified number of pixels.

SYNTAX

```
let result = Action.ExpandSelection(expandByPixels);
```

PARAMETERS

Parameter	Description	Default
`expandByPixels` Integer, Optional	Expansion distance in pixels.	5

RETURN VALUE

Boolean — True if the selection changed; otherwise False.

EXAMPLE CODE

```
let success = Action.ExpandSelection(8);
```

5. Action.GetFromImageBuffer(sourceLocation) Returns: Boolean

DESCRIPTION

Loads an image from an image-buffer location into the workspace.

SYNTAX

```
let result = Action.GetFromImageBuffer(sourceLocation);
```

PARAMETERS

Parameter	Description	Default
`sourceLocation` Integer, Optional	Source buffer index from 1 to 10.	1

RETURN VALUE

Boolean — True if the image was loaded; otherwise False.

EXAMPLE CODE

```
let success = Action.GetFromImageBuffer(1);
```

6. Action.Paste() Returns: Boolean

DESCRIPTION

Pastes an image from the clipboard into the workspace.

SYNTAX

```
let result = Action.Paste();
```

RETURN VALUE

Boolean — True if an image was pasted; otherwise False.

EXAMPLE CODE

```
let success = Action.Paste();
```

7. Action.PasteAndFitToRectangle() Returns: Boolean

DESCRIPTION

Pastes an image from the clipboard and fits it to the current rectangle.

SYNTAX

```
let result = Action.PasteAndFitToRectangle();
```

RETURN VALUE

Boolean — True if an image was pasted; otherwise False.

EXAMPLE CODE

```
let success = Action.PasteAndFitToRectangle();
```

8. Action.PasteSelectionAsBinaryToBuffer(targetBufferIndex)

Returns: Boolean

DESCRIPTION

Exports the current selection mask as a binary image to a buffer.

SYNTAX

```
let result = Action.PasteSelectionAsBinaryToBuffer(targetBufferIndex);
```

PARAMETERS

Parameter	Description	Default
`targetBufferIndex` Integer, Optional	Target buffer index from 1 to 10.	1

RETURN VALUE

Boolean — True if the mask was written; otherwise False.

EXAMPLE CODE

```
let success = Action.PasteSelectionAsBinaryToBuffer(1);
```

9. Action.PutToImageBuffer(targetLocation) Returns: Boolean

DESCRIPTION

Copies the current image to an image-buffer location.

SYNTAX

```
let result = Action.PutToImageBuffer(targetLocation);
```

PARAMETERS

Parameter	Description	Default

`targetLocation` Integer, Optional	Target buffer index from 1 to 10.	1

RETURN VALUE

Boolean — True if the image was stored; otherwise False.

EXAMPLE CODE

```
let success = Action.PutToImageBuffer(1);
```

10. Action.ReverseSelection() Returns: Boolean

DESCRIPTION

Inverts the current selection mask.

SYNTAX

```
let result = Action.ReverseSelection();
```

RETURN VALUE

Boolean — True if the selection was reversed; otherwise False.

EXAMPLE CODE

```
let success = Action.ReverseSelection();
```

11. Action.ShrinkSelection(shrinkByPixels) Returns: Boolean

DESCRIPTION

Shrinks the current selection by a specified number of pixels.

SYNTAX

```
let result = Action.ShrinkSelection(shrinkByPixels);
```

PARAMETERS

Parameter	Description	Default
`shrinkByPixels` Integer, Optional	Shrink distance in pixels.	5

RETURN VALUE

Boolean — True if the selection changed; otherwise False.

EXAMPLE CODE

```
let success = Action.ShrinkSelection(3);
```

12. Action.SubtractSelectionFromBinaryImage(sourceImageLocation)

Returns: Boolean

DESCRIPTION

Subtracts a binary image mask from the current selection using logical AND NOT.

SYNTAX

```
let result = Action.SubtractSelectionFromBinaryImage(sourceImageLocation);
```

PARAMETERS

Parameter	Description	Default
`sourceImageLocation` Integer, Optional	Source buffer index from 1 to 10.	1

RETURN VALUE

Boolean — True if the selection was updated; otherwise False.

EXAMPLE CODE

```
let success = Action.SubtractSelectionFromBinaryImage(1);
```

13. Action.Unselect() Returns: Boolean

DESCRIPTION

Clears the current selection.

SYNTAX

```
let result = Action.Unselect();
```

RETURN VALUE

Boolean — True if the selection was cleared; otherwise False.

EXAMPLE CODE

```
let success = Action.Unselect();
```

Hardware & Capture Actions 2 functions

1. Action.AcquireImagefromScanner(useSilentMode) Returns: Boolean

DESCRIPTION

Acquires an image from a scanner.

SYNTAX

```
let result = Action.AcquireImagefromScanner(useSilentMode);
```

PARAMETERS

Parameter	Description	Default
`useSilentMode` Boolean, Optional	Whether to acquire without showing the scanner UI.	True

RETURN VALUE

Boolean — True if acquisition succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.AcquireImagefromScanner(true);
```

2. Action.Capture() Returns: Boolean

DESCRIPTION

Captures an image from a connected camera or scanner.

SYNTAX

```
let result = Action.Capture();
```

RETURN VALUE

Boolean — True if capture succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.Capture();
```

Image Transforms & Filters 32 functions

1. Action.AdvancedSharpening(sharpenAmountPercentage)

Returns: Boolean

DESCRIPTION

Sharpens the image using the advanced sharpening filter.

SYNTAX

```
let result = Action.AdvancedSharpening(sharpenAmountPercentage);
```

PARAMETERS

Parameter	Description	Default
`sharpenAmountPercentage` Integer, Optional	Sharpening amount percentage.	100

RETURN VALUE

Boolean — True if sharpening was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.AdvancedSharpening(125);
```

2. Action.BoxAveraging(kernelSize) Returns: Boolean

DESCRIPTION

Smooths the image using a box-averaging filter.

SYNTAX

```
let result = Action.BoxAveraging(kernelSize);
```

PARAMETERS

Parameter	Description	Default
`kernelSize` Integer, Optional	Averaging kernel width and height in pixels.	3

RETURN VALUE

Boolean — True if the filter was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.BoxAveraging(5);
```

3. Action.ChromaKeyOverlay(sourceLocation1, targetColorRed, targetColorGreen, targetColorBlue, sourceLocation2, opacity, enableFeathering, featheringRadius, tolerance, softness)

Returns: Boolean

DESCRIPTION

Overlays an image while making the selected chroma-key color transparent.

SYNTAX

```
let result = Action.ChromaKeyOverlay(sourceLocation1, targetColorRed, targetColorGreen, targetColorBlue, sourceLocation2, opacity, enableFeathering, featheringRadius, tolerance, softness);
```

PARAMETERS

Parameter	Description	Default
`sourceLocation1` Integer, Optional	Foreground image source (0 for workspace, 1–10 for buffers).	1
`targetColorRed` Integer, Optional	Red component of the chroma-key color (0–255).	0
`targetColorGreen` Integer, Optional	Green component of the chroma-key color (0–255).	0
`targetColorBlue` Integer, Optional	Blue component of the chroma-key color (0–255).	0
`sourceLocation2` Integer, Optional	Background image source (0 for workspace, 1–10 for buffers).	0
`opacity` Integer, Optional	Overlay opacity percentage (0–100).	100
`enableFeathering` Boolean, Optional	Whether to soften the keyed edge.	False
`featheringRadius` Integer, Optional	Edge feathering radius in pixels.	30
`tolerance` Integer, Optional	Color matching tolerance range (0–100).	35
`softness` Integer, Optional	Edge gradient softness range (0–100).	20

RETURN VALUE

Boolean — True if the overlay was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.ChromaKeyOverlay(1, 0, 255, 0, 0, 100, false, 30, 35, 20);
```

4. Action.Crop(cropWidthPixels, cropHeightPixels, startXCoordinate, startYCoordinate)

Returns: Boolean

DESCRIPTION

Crops a rectangular area from the image.

SYNTAX

```
let result = Action.Crop(cropWidthPixels, cropHeightPixels, startXCoordinate, startYCoordinate);
```

PARAMETERS

Parameter	Description	Default
`cropWidthPixels` Integer, Optional	Crop width in pixels.	500
`cropHeightPixels` Integer, Optional	Crop height in pixels.	500
`startXCoordinate` Integer, Optional	Left coordinate of the crop in pixels.	0
`startYCoordinate` Integer, Optional	Top coordinate of the crop in pixels.	0

RETURN VALUE

Boolean — True if cropping was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.Crop(500, 500, 0, 0);
```

5. Action.Dehaze(patchRadius, dehazeStrength, saturationBoost,

brightnessBoost)

Returns: Boolean

DESCRIPTION

Removes atmospheric haze, fog, and chamber glare to restore crisp contrast and vivid colors using a high-speed O(1) Dark Channel Prior algorithm.

SYNTAX

```
let result = Action.Dehaze(patchRadius, dehazeStrength, saturatio
nBoost, brightnessBoost);
```

PARAMETERS

Parameter	Description	Default
`patchRadius` Integer, Optional	Local patch radius in pixels for haze estimation. Range: 1 to 150.	25
`dehazeStrength` Single, Optional	Haze removal intensity. Range: 0.0 to 1.0.	0.7F
`saturationBoost` Single, Optional	Color saturation boost after haze removal. Range: -1.0 to 1.0.	0.15F
`brightnessBoost` Integer, Optional	Additional brightness compensation. Range: -100 to 100.	10

RETURN VALUE

Boolean — True if the Dehaze filter was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.Dehaze(25, 0.7F, 0.15F, 10);
```

6. Action.Dithering(ditheringAlgorithmType) Returns: Boolean

DESCRIPTION

Applies a dithering algorithm to reduce image color depth.

SYNTAX

```
let result = Action.Dithering(ditheringAlgorithmType);
```

PARAMETERS

Parameter	Description	Default

ditheringAlgorithmType Integer, Optional	Integer representing the dithering algorithm. Supported ditheringAlgorithmType values: 0: Ordered 1: FloydSteinberg	0

RETURN VALUE

Boolean — True if dithering was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.Dithering(0);
```

7. Action.Emboss(angle, distance, strength, bias) Returns: Boolean

DESCRIPTION

Creates an embossed relief effect.

SYNTAX

```
let result = Action.Emboss(angle, distance, strength, bias);
```

PARAMETERS

Parameter	Description	Default
angle Single, Optional	Light-source angle in degrees.	135.0F
distance Integer, Optional	Relief offset in pixels.	1
strength Integer, Optional	Effect strength percentage.	100
bias Integer, Optional	Output intensity bias.	128

RETURN VALUE

Boolean — True if the effect was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.Emboss(135.0F, 1, 100, 128);
```

8. Action.FFT() Returns: Boolean

DESCRIPTION

Transforms the image into the frequency domain using FFT.

SYNTAX

```
let result = Action.FFT();
```

RETURN VALUE

Boolean — True if the transform was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.FFT();
```

9. Action.FlipHorizontally() Returns: Boolean

DESCRIPTION

Flips the image horizontally.

SYNTAX

```
let result = Action.FlipHorizontally();
```

RETURN VALUE

Boolean — True if the image was transformed; otherwise False.

EXAMPLE CODE

```
let success = Action.FlipHorizontally();
```

10. Action.FlipVertically() Returns: Boolean

DESCRIPTION

Flips the image vertically.

SYNTAX

```
let result = Action.FlipVertically();
```

RETURN VALUE

Boolean — True if the image was transformed; otherwise False.

EXAMPLE CODE

```
let success = Action.FlipVertically();
```

```
let success = Action.FlipVertically();
```

11. Action.GammaAdjustment(gammaValue) Returns: Boolean

DESCRIPTION

Adjusts image brightness using gamma correction.

SYNTAX

```
let result = Action.GammaAdjustment(gammaValue);
```

PARAMETERS

Parameter	Description	Default
`gammaValue` Double, Optional	Gamma correction value.	0.1

RETURN VALUE

Boolean — True if the adjustment was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.GammaAdjustment(0.8);
```

12. Action.GaussianFilter(kernelSize) Returns: Boolean

DESCRIPTION

Blurs the image using a Gaussian filter.

SYNTAX

```
let result = Action.GaussianFilter(kernelSize);
```

PARAMETERS

Parameter	Description	Default
`kernelSize` Integer, Optional	Gaussian kernel width and height in pixels.	3

RETURN VALUE

Boolean — True if the filter was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.GaussianFilter(5);
```

13. Action.HDRLocalToneMapping(blurRadius, detailStrength, toneCompression, brightnessBoost)

Returns: Boolean

DESCRIPTION

Enhances local contrast and compresses dynamic range (shadow/highlight recovery) using a high-speed O(1) Stack Blur algorithm.

SYNTAX

```
let result = Action.HDRLocalToneMapping(blurRadius, detailStrength, toneCompression, brightnessBoost);
```

PARAMETERS

Parameter	Description	Default
blurRadius Integer, Optional	Local neighborhood radius in pixels. Range: 1 to 150.	40
detailStrength Single, Optional	Clarity and fine-detail amplification strength. Range: 0.0 to 3.0.	1.2F
toneCompression Single, Optional	Shadow/highlight dynamic range compression strength. Range: 0.0 to 1.0.	0.5F
brightnessBoost Integer, Optional	Additional brightness boost applied to the final pixels. Range: -100 to 100.	0

RETURN VALUE

Boolean — True if the HDR Local Tone Mapping filter was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.HDRLocalToneMapping(40, 1.2F, 0.5F, 0);
```

14. Action.HighlightBloomHalation(spreadRadius, threshold, bloomStrength, halationTint)

Returns: Boolean

DESCRIPTION

Applies selective optical highlight bloom and analog cinema film halation to bright light sources using a high-speed O(1) Stack Blur engine.

SYNTAX

```
let result = Action.HighlightBloomHalation(spreadRadius, threshold, bloomStrength, halationTint);
```

PARAMETERS

Parameter	Description	Default
spreadRadius Integer, Optional	Diffusion radius in pixels. Range: 1 to 150.	25
threshold Integer, Optional	Luminance threshold for isolating bright highlights. Range: 0 to 250.	180
bloomStrength Single, Optional	Intensity of the highlight bloom glow. Range: 0.0 to 2.0.	0.8F
halationTint Single, Optional	Red-orange cinema film halation tint ratio (0.0 for pure optical bloom, 1.0 for full film halation). Range: 0.0 to 1.0.	0.0F

RETURN VALUE

Boolean — True if the Highlight Bloom & Halation filter was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.HighlightBloomHalation(25, 180, 0.8F, 0.35F);
```

15. Action.HighPassSharpening(radius, strength, threshold, blendMode)

Returns: Boolean

DESCRIPTION

Sharpens multi-scale structural edges and surface textures using a high-speed O(1) High-Pass frequency extraction algorithm.

SYNTAX

```
let result = Action.HighPassSharpening(radius, strength, threshold, blendMode);
```

PARAMETERS

Parameter	Description	Default
radius	Frequency radius in pixels. Range: 1 to	10

`radius` Integer, Optional	Frequency radius in pixels. Range: 1 to 100.	10
`strength` Single, Optional	High-frequency detail amplification strength. Range: 0.0 to 2.0.	1.0F
`threshold` Integer, Optional	Noise suppression threshold in pixel levels. Range: 0 to 50.	3
`blendMode` Integer, Optional	Integer representing the blending algorithm. Supported blendMode values: 0: Overlay (Balanced) 1: SoftLight (Subtle) 2: HardLight (Intense)	0

RETURN VALUE

Boolean — True if the High-Pass Sharpening filter was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.HighPassSharpening(10, 1.0F, 3, 0);
```

16. Action.ImageStamp(stampImageSource, locationX, locationY, scaleX, scaleY, angle, opacity, enableTransparentColor, transparentColorArgb, enableGradualAlpha, tolerance, enableFeather, featherRadius)

Returns: Boolean

DESCRIPTION

Places and transforms a buffer image stamp over the current image.

SYNTAX

```
let result = Action.ImageStamp(stampImageSource, locationX, locationY, scaleX, scaleY, angle, opacity, enableTransparentColor, transparentColorArgb, enableGradualAlpha, tolerance, enableFeather, featherRadius);
```

PARAMETERS

Parameter	Description	Default
`stampImageSource` Integer, Optional	Stamp image source buffer index.	1
`locationX` Integer, Optional	Stamp left coordinate in pixels.	0

locationY Integer, Optional	Stamp top coordinate in pixels.	0
scaleX Integer, Optional	Horizontal scale percentage.	100
scaleY Integer, Optional	Vertical scale percentage.	100
angle Single, Optional	Rotation angle in degrees.	0.0F
opacity Integer, Optional	Stamp opacity percentage.	100
enableTransparentColor Boolean, Optional	Whether to make a selected color transparent.	False
transparentColorArgb Integer, Optional	ARGB value of the transparent color.	-1
enableGradualAlpha Boolean, Optional	Whether to apply gradual alpha blending.	False
tolerance Integer, Optional	Transparent-color matching tolerance.	30
enableFeather Boolean, Optional	Whether to feather stamp edges.	False
featherRadius Integer, Optional	Feathering radius in pixels.	10

RETURN VALUE

Boolean — True if the stamp was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.ImageStamp(1, 0, 0, 100, 100, 0.0F, 100,
false, -1, false, 30, false, 10);
```

17. Action.InverseFFT() Returns: Boolean

DESCRIPTION

Reconstructs a spatial image from frequency-domain data using inverse FFT.

SYNTAX

```
let result = Action.InverseFFT();
```

RETURN VALUE

Boolean — True if the inverse transform was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.InverseFFT();
```

18. Action.MiniatureEffect(blurRadius, focusPosition, focusSize, saturationBoost)

Returns: Boolean

DESCRIPTION

Simulates a tilt-shift miniature lens effect with smooth depth-of-field blurring and saturation enhancement.

SYNTAX

```
let result = Action.MiniatureEffect(blurRadius, focusPosition, focusSize, saturationBoost);
```

PARAMETERS

Parameter	Description	Default
blurRadius Integer, Optional	Out-of-focus blur radius in pixels. Range: 1 to 150.	25
focusPosition Single, Optional	Vertical center position of the sharp focus zone (0.0 = top, 0.5 = center, 1.0 = bottom). Range: 0.0 to 1.0.	0.5F
focusSize Single, Optional	Relative height of the sharp focus band. Range: 0.05 to 0.80.	0.25F
saturationBoost Single, Optional	Color vibrancy boost for a miniature model look. Range: -1.0 to 1.0.	0.25F

RETURN VALUE

Boolean — True if the Miniature Effect filter was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.MiniatureEffect(25, 0.5F, 0.25F, 0.25F);
```

19. Action.MosaicFilter(mosaicModeType, value) Returns: Boolean

DESCRIPTION

Applies a mosaic or pixelation effect.

SYNTAX

```
let result = Action.MosaicFilter(mosaicModeType, value);
```

PARAMETERS

Parameter	Description	Default
mosaicModeType Integer, Optional	Integer representing the mosaic mode. Supported mosaicModeType values: 0: PixelSize - value is the block size in pixels 1: GridCount - value is the number of grid cells	0
value Integer, Optional	Mosaic block size or mode-specific amount.	10

RETURN VALUE

Boolean — True if the effect was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.MosaicFilter(0, 10);
```

20. Action.OrtonEffect(blurRadius, glowStrength, exposureBoost, brightnessBoost)

Returns: Boolean

DESCRIPTION

Applies a dreamy, painterly luminescence (Orton Effect) by combining overexposed sharp and blurred layers via Multiply blending.

SYNTAX

```
let result = Action.OrtonEffect(blurRadius, glowStrength, exposureBoost, brightnessBoost);
```

PARAMETERS

Parameter	Description	Default
`blurRadius` Integer, Optional	Blur radius for the glow diffusion in pixels. Range: 1 to 150.	25
`glowStrength` Single, Optional	Overall blending opacity of the Orton effect. Range: 0.0 to 1.0.	0.6F
`exposureBoost` Single, Optional	Highlight overexposure boost in the Screen pass. Range: 0.0 to 1.0.	0.7F
`brightnessBoost` Integer, Optional	Additional brightness boost applied to the final pixels. Range: -100 to 100.	0

RETURN VALUE

Boolean — True if the Orton Effect filter was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.OrtonEffect(25, 0.6F, 0.7F, 0);
```

21. Action.OverlayImage(sourceLocation1, sourceLocation2, targetColorRed, targetColorGreen, targetColorBlue, opacity, enableFeathering, featheringRadius, tolerance, gradientAlpha)

Returns: Boolean

DESCRIPTION

Overlays two workspace or buffer images with opacity, blending, and feathering options.

SYNTAX

```
let result = Action.OverlayImage(sourceLocation1, sourceLocation2, targetColorRed, targetColorGreen, targetColorBlue, opacity, enableFeathering, featheringRadius, tolerance, gradientAlpha);
```

PARAMETERS

Parameter	Description	Default
`sourceLocation1` Integer, Optional	First image source (0 for workspace, 1–10 for buffers).	1
`sourceLocation2` Integer, Optional	Second image source (0 for workspace, 1–10 for buffers).	0
`targetColorRed` Integer, Optional	Red component of the target color (0–255).	0

`targetColorGreen` Integer, Optional	Green component of the target color (0–255).	0
`targetColorBlue` Integer, Optional	Blue component of the target color (0–255).	0
`opacity` Integer, Optional	Overlay opacity percentage.	100
`enableFeathering` Boolean, Optional	Whether to feather the overlay edge.	False
`featheringRadius` Integer, Optional	Feathering radius in pixels.	30
`tolerance` Integer, Optional	Color matching tolerance.	0
`gradientAlpha` Boolean, Optional	Whether to apply gradual alpha blending.	False

RETURN VALUE

Boolean — True if the overlay was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.OverlayImage(1, 0, 0, 0, 0, 100, false, 30, 0, false);
```

22. Action.Resize(resizeUnitType, maintainAspectRatio, baseDimensionType, targetWidth, targetHeight, interpolationAlgorithmType)

Returns: Boolean

DESCRIPTION

Resizes the image to the requested dimensions using the selected interpolation.

SYNTAX

```
let result = Action.Resize(resizeUnitType, maintainAspectRatio, baseDimensionType, targetWidth, targetHeight, interpolationAlgorithmType);
```

PARAMETERS

Parameter	Description	Default

resizeUnitType Integer, Optional	Integer representing the resize units. Supported resizeUnitType values: 0: Pixels 1: Percent	0
maintainAspectRatio Boolean, Optional	Whether to preserve the image aspect ratio.	True
baseDimensionType Integer, Optional	Integer representing the dimension basis. Supported baseDimensionType values: 0: Width 1: Height	0
targetWidth Integer, Optional	Requested width.	800
targetHeight Integer, Optional	Requested height.	600
interpolationAlgorithmType Integer, Optional	Integer representing the interpolation algorithm. Supported interpolationAlgorithmType values: 0: NearestNeighbor 1: Bilinear 2: Bicubic 3: HighQualityBilinear 4: HighQualityBicubic	4

RETURN VALUE

Boolean — True if the image was resized; otherwise False.

EXAMPLE CODE

```
let success = Action.Resize(0, true, 0, 800, 600, 4);
```

23. Action.ResizeCanvas(resizeUnitType, maintainAspectRatio, baseDimensionType, targetWidth, targetHeight, canvasAlignmentType, backgroundColorRed, backgroundColorGreen, backgroundColorBlue)

Returns: Boolean

Returns: Boolean

DESCRIPTION

Resizes the canvas around the image with alignment and fill-color options.

SYNTAX

```
let result = Action.ResizeCanvas(resizeUnitType, maintainAspectRatio, baseDimensionType, targetWidth, targetHeight, canvasAlignmentType, backgroundColorRed, backgroundColorGreen, backgroundColorBlue);
```

PARAMETERS

Parameter	Description	Default
`resizeUnitType` Integer, Optional	Integer representing the resize units. Supported resizeUnitType values: 0: Pixels 1: Percent	0
`maintainAspectRatio` Boolean, Optional	Whether to preserve the image aspect ratio.	True
`baseDimensionType` Integer, Optional	Integer representing the dimension basis. Supported baseDimensionType values: 0: Width 1: Height	0
`targetWidth` Integer, Optional	Requested canvas width.	800
`targetHeight` Integer, Optional	Requested canvas height.	600
`canvasAlignmentType` Integer, Optional	Integer representing the image alignment. Supported canvasAlignmentType values: 0: Center 1: LeftTop 2: RightTop 3: LeftBottom 4: RightBottom	0
`backgroundColorRed` Integer, Optional	Red component of the canvas fill color (0–255).	255

backgroundColorGreen Integer, Optional	Green component of the canvas fill color (0–255).	255
backgroundColorBlue Integer, Optional	Blue component of the canvas fill color (0–255).	255

RETURN VALUE

Boolean — True if the canvas was resized; otherwise False.

EXAMPLE CODE

```
let success = Action.ResizeCanvas(0, true, 0, 800, 600, 0, 255, 255, 255);
```

24. Action.Rotate(rotationAngleDegrees, rotationDirectionType, backgroundColorRed, backgroundColorGreen, backgroundColorBlue)

Returns: Boolean

DESCRIPTION

Rotates the image by the requested angle and fills uncovered areas.

SYNTAX

```
let result = Action.Rotate(rotationAngleDegrees, rotationDirectionType, backgroundColorRed, backgroundColorGreen, backgroundColorBlue);
```

PARAMETERS

Parameter	Description	Default
rotationAngleDegrees Single, Optional	Rotation angle in degrees.	90.0F
rotationDirectionType Integer, Optional	Integer representing the rotation direction. Supported rotationDirectionType values: 0: CounterClockwise 1: Clockwise	1
backgroundColorRed Integer, Optional	Red component of the fill color (0–255).	255
backgroundColorGreen Integer, Optional	Green component of the fill color (0–255).	255

backgroundColorBlue Integer, Optional	Blue component of the fill color (0–255).	255

RETURN VALUE

Boolean — True if rotation was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.Rotate(90.0F, 1, 255, 255, 255);
```

25. Action.RotationalAlignmentByLine(backgroundColorRed, backgroundColorGreen, backgroundColorBlue)

Returns: Boolean

DESCRIPTION

Rotates the image to align it with a detected line.

SYNTAX

```
let result = Action.RotationalAlignmentByLine(backgroundColorRed, backgroundColorGreen, backgroundColorBlue);
```

PARAMETERS

Parameter	Description	Default
backgroundColorRed Integer, Optional	Red component of the fill color (0–255).	255
backgroundColorGreen Integer, Optional	Green component of the fill color (0–255).	255
backgroundColorBlue Integer, Optional	Blue component of the fill color (0–255).	255

RETURN VALUE

Boolean — True if alignment was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.RotationalAlignmentByLine(255, 255, 255);
```

26. Action.Sharpen3x3() Returns: Boolean

DESCRIPTION

Sharpens image details using a 3×3 convolution filter.

SYNTAX

```
let result = Action.Sharpen3x3();
```

RETURN VALUE

Boolean — True if sharpening was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.Sharpen3x3();
```

27. Action.Sharpen5x5HighQuality(sharpenStrengthPercentage)

Returns: Boolean

DESCRIPTION

Sharpens the image using a high-quality 5×5 filter.

SYNTAX

```
let result = Action.Sharpen5x5HighQuality(sharpenStrengthPercentage);
```

PARAMETERS

Parameter	Description	Default
`sharpenStrengthPercentage` Integer, Optional	Sharpening strength percentage.	10

RETURN VALUE

Boolean — True if sharpening was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.Sharpen5x5HighQuality(20);
```

28. Action.SmartDenoise(radius, threshold) Returns: Boolean

DESCRIPTION

Reduces image noise while preserving edges.

SYNTAX

```
let result = Action.SmartDenoise(radius, threshold);
```

PARAMETERS

Parameter	Description	Default
`radius` Integer, Optional	Denoising neighborhood radius.	5
`threshold` Integer, Optional	Noise similarity threshold.	30

RETURN VALUE

Boolean — True if denoising was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.SmartDenoise(5, 30);
```

29. Action.SmartSketch(strokeWidth, pencilDarkness, colorBlend, paperBrightness)

Returns: Boolean

DESCRIPTION

Converts the image into a fine pencil sketch or colored-pencil technical line art using a high-speed O(1) Color Dodge algorithm.

SYNTAX

```
let result = Action.SmartSketch(strokeWidth, pencilDarkness, colorBlend, paperBrightness);
```

PARAMETERS

Parameter	Description	Default
`strokeWidth` Integer, Optional	Stroke width (blur radius) in pixels. Range: 1 to 100.	15
`pencilDarkness` Single, Optional	Graphite line darkness and shading intensity. Range: 0.01 to 1.0.	0.5F
`colorBlend` Single, Optional	Original color blending ratio (0.0 for pure grayscale sketch, up to 1.0 for colored-pencil look). Range: 0.0 to 1.0.	0.0F
`paperBrightness` Integer, Optional	Background paper brightness adjustment. Range: -100 to 100.	0

RETURN VALUE

Boolean — True if the Smart Sketch filter was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.SmartSketch(15, 0.5F, 0.0F, 0);
```

30. Action.SoftGlow(blurRadius, glowStrength, brightnessBoost, blendMode)

Returns: Boolean

DESCRIPTION

Applies a dreamy, radiant luminescence effect using a high-speed O(1) Stack Blur and Screen/SoftLight blending.

SYNTAX

```
let result = Action.SoftGlow(blurRadius, glowStrength, brightness
Boost, blendMode);
```

PARAMETERS

Parameter	Description	Default
blurRadius Integer, Optional	Blur radius for the glow diffusion in pixels. Range: 1 to 150.	20
glowStrength Single, Optional	Blending strength (opacity) of the glow layer. Range: 0.0 to 1.0.	0.45F
brightnessBoost Integer, Optional	Additional brightness boost applied to the final blended pixels. Range: -100 to 100.	5
blendMode Integer, Optional	Integer representing the blending algorithm used to combine the glow layer with the base image. Supported blendMode values: 0: Screen (Bright & Airy) 1: SoftLight (Gentle Contrast)	0

RETURN VALUE

Boolean — True if the Soft Glow filter was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.SoftGlow(20, 0.45F, 5, 0);
```

31. Action.StackBlur(kernelSize) Returns: Boolean

DESCRIPTION

Blurs the image using a high-speed Stack Blur algorithm.

SYNTAX

```
let result = Action.StackBlur(kernelSize);
```

PARAMETERS

Parameter	Description	Default
kernelSize Integer, Optional	Blur radius (half size of the kernel) in pixels. Range: 1 to 100.	3

RETURN VALUE

Boolean — True if the filter was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.StackBlur(5);
```

32. Action.TextAnnotation(textString, locationX, locationY, fontName, fontSize, fontStyleInt, innerColorArgb, outerColorArgb, opacity, outlineThickness, angle, isAntiAlias)

Returns: Boolean

DESCRIPTION

Adds a styled text annotation or watermark to the image.

SYNTAX

```
let result = Action.TextAnnotation(textString, locationX, locatio
nY, fontName, fontSize, fontStyleInt, innerColorArgb, outerColorA
rgb, opacity, outlineThickness, angle, isAntiAlias);
```

PARAMETERS

Parameter	Description	Default
textString String, Optional	Text to draw.	"Text"
locationX	Text left coordinate in pixels.	0

Integer, Optional		
locationY Integer, Optional	Text top coordinate in pixels.	0
fontName String, Optional	Font family name.	"Arial"
fontSize Single, Optional	Font size in points.	12.0F
fontStyleInt Integer, Optional	Integer representing the font style. Supported fontStyleInt values: 0: Regular 1: Bold 2: Italic 3: Bold Italic 4: Underline 8: Strikeout	0
innerColorArgb Integer, Optional	ARGB value of the text fill color.	-256
outerColorArgb Integer, Optional	ARGB value of the outline color.	-16777216
opacity Integer, Optional	Text opacity percentage.	100
outlineThickness Single, Optional	Outline thickness in pixels.	1.0F
angle Single, Optional	Text rotation angle in degrees.	0.0F
isAntiAlias Boolean, Optional	Whether to antialias the text.	True

RETURN VALUE

Boolean — True if the annotation was drawn; otherwise False.

EXAMPLE CODE

```
let success = Action.TextAnnotation("Sample", 20, 20, "Arial",
12.0F, 0, -256, -16777216, 100, 1.0F, 0.0F, true);
```

Neural & AI Processing Actions 23 functions

1. Action.AIAutoSubjectSelectionGeneral() Returns: Boolean

DESCRIPTION

Automatically selects the main subject in a general image.

SYNTAX

```
let result = Action.AIAutoSubjectSelectionGeneral();
```

RETURN VALUE

Boolean — True if subject selection succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.AIAutoSubjectSelectionGeneral();
```

2. Action.AIAutoSubjectSelectionPortrait() Returns: Boolean

DESCRIPTION

Automatically selects the subject in a portrait image.

SYNTAX

```
let result = Action.AIAutoSubjectSelectionPortrait();
```

RETURN VALUE

Boolean — True if subject selection succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.AIAutoSubjectSelectionPortrait();
```

3. Action.AIBackgroundRemovalGeneral(virtualBackgroundColorRed, virtualBackgroundColorGreen, virtualBackgroundColorBlue)

Returns: Boolean

DESCRIPTION

Removes the background from a general image and fills it with a color.

SYNTAX

```
let result = Action.AIBackgroundRemovalGeneral(virtualBackgroundC
olorRed, virtualBackgroundColorGreen, virtualBackgroundColorBlu
```

```
e);
```

PARAMETERS

Parameter	Description	Default
`virtualBackgroundColorRed` Integer, Optional	Replacement background red component (0–255).	255
`virtualBackgroundColorGreen` Integer, Optional	Replacement background green component (0–255).	255
`virtualBackgroundColorBlue` Integer, Optional	Replacement background blue component (0–255).	255

RETURN VALUE

Boolean — True if background removal succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.AIBackgroundRemovalGeneral(255, 255, 255);
```

4. Action.AIBackgroundRemovalPortrait(virtualBackgroundColorRed, virtualBackgroundColorGreen, virtualBackgroundColorBlue)

Returns: Boolean

DESCRIPTION

Removes the background from a portrait image and fills it with a color.

SYNTAX

```
let result = Action.AIBackgroundRemovalPortrait(virtualBackgroundColorRed, virtualBackgroundColorGreen, virtualBackgroundColorBlue);
```

PARAMETERS

Parameter	Description	Default
`virtualBackgroundColorRed` Integer, Optional	Replacement background red component (0–255).	255
`virtualBackgroundColorGreen` Integer, Optional	Replacement background green component (0–255).	255
`virtualBackgroundColorBlue`	Replacement background	255

	Integer, Optional	blue component (0–255).

RETURN VALUE

Boolean — True if background removal succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.AIBackgroundRemovalPortrait(255, 255, 255);
```

5. Action.AIBackgroundRemovalSelfie(virtualBackgroundColorRed, virtualBackgroundColorGreen, virtualBackgroundColorBlue)

Returns: Boolean

DESCRIPTION

Removes the background from a selfie and fills it with a color.

SYNTAX

```
let result = Action.AIBackgroundRemovalSelfie(virtualBackgroundColorRed, virtualBackgroundColorGreen, virtualBackgroundColorBlue);
```

PARAMETERS

Parameter	Description	Default
`virtualBackgroundColorRed` Integer, Optional	Replacement background red component (0–255).	255
`virtualBackgroundColorGreen` Integer, Optional	Replacement background green component (0–255).	255
`virtualBackgroundColorBlue` Integer, Optional	Replacement background blue component (0–255).	255

RETURN VALUE

Boolean — True if background removal succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.AIBackgroundRemovalSelfie(255, 255, 255);
```

6. Action.AIBinarizationSegmentation() Returns: Boolean

DESCRIPTION

Segments cells or nuclei in an image and binarizes them using the StarDist AI model.

SYNTAX

```
let result = Action.AIBinarizationSegmentation();
```

RETURN VALUE

Boolean — True if StarDist segmentation and binarization succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.AIBinarizationSegmentationStarDist();
```

7. Action.AICustomBinarization(targetModelName) Returns: Boolean

DESCRIPTION

Binarizes the image using a selected custom ONNX model.

SYNTAX

```
let result = Action.AICustomBinarization(targetModelName);
```

PARAMETERS

Parameter	Description	Default
`targetModelName` String, Optional	Model name to use; an empty string selects the default model.	""

RETURN VALUE

Boolean — True if binarization succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.AICustomBinarization("");
```

8. Action.AIDepthEstimation(autoResizeLargeImages) Returns: Boolean

DESCRIPTION

Estimates an image depth map using an AI model.

SYNTAX

```
let result = Action.AIDepthEstimation(autoResizeLargeImages);
```

PARAMETERS

Parameter	Description	Default
`autoResizeLargeImages` Boolean, Optional	Whether to resize large images before inference.	True

RETURN VALUE

Boolean — True if depth estimation succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.AIDepthEstimation(true);
```

9. Action.AIDocumentUnwarping() Returns: Boolean

DESCRIPTION

Straightens and unwarps a perspective-distorted document image.

SYNTAX

```
let result = Action.AIDocumentUnwarping();
```

RETURN VALUE

Boolean — True if document unwarping succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.AIDocumentUnwarping();
```

10. Action.AIFaceDetection(confidenceThreshold, nmsThreshold, enableTiling, useRotatedEllipse, applyToMask, drawBoundingBoxes, showFaceID, showConfidence, showKeypoints, boxColorRed, boxColorGreen, boxColorBlue, exportCSV, csvExportDirectory)

Returns: Boolean

DESCRIPTION

Detects faces and 6-point landmarks using an AI model, and optionally applies an ROI selection mask, draws visual overlays, or exports a CSV.

SYNTAX

```
let result = Action.AIFaceDetection(confidenceThreshold, nmsThres
hold, enableTiling, useRotatedEllipse, applyToMask, drawBoundingB
oxes, showFaceID, showConfidence, showKeypoints, boxColorRed, box
```

```
colorGreen, boxColorBlue, exportCSV, csvExportDirectory);
```

PARAMETERS

Parameter	Description	Default
confidenceThreshold Single, Optional	Minimum face detection confidence (0.0–1.0).	0.45F
nmsThreshold Single, Optional	Non-maximum suppression threshold (0.0–1.0).	0.4F
enableTiling Boolean, Optional	Whether to process the image in multi-scale tiles for high-res/group photos.	True
useRotatedEllipse Boolean, Optional	True to use eye-aligned rotated ellipses; False for rectangles.	True
applyToMask Boolean, Optional	True to convert detected faces into an active workspace selection mask (AdvSel) for subsequent filtering/blurring.	False
drawBoundingBoxes Boolean, Optional	Whether to draw the bounding ROI shape (ellipse or rectangle).	True
showFaceID Boolean, Optional	Whether to display the sequential face ID (#1, #2...).	True
showConfidence Boolean, Optional	Whether to display confidence scores.	False
showKeypoints Boolean, Optional	Whether to draw the 6 facial landmarks.	True
boxColorRed Integer, Optional	Bounding-box red component (0–255).	255
boxColorGreen Integer, Optional	Bounding-box green component (0–255).	0
boxColorBlue Integer, Optional	Bounding-box blue component (0–255).	0
exportCSV Boolean, Optional	Whether to export face ROI and 6-keypoint coordinates as CSV.	False
csvExportDirectory String, Optional	Directory for CSV output; empty uses the source image folder.	""

RETURN VALUE

Boolean — True if face detection completed; otherwise False.

EXAMPLE CODE

```
let success = Action.AIFaceDetection(0.45F, 0.4F, true, true,
false, true, true, false, true, 255, 0, 0, false, "");
```

11. Action.AIImageQualityRestorationFast(restorationBoostFactor)

Returns: Boolean

DESCRIPTION

Restores a low-quality image using the fast AI model.

SYNTAX

```
let result = Action.AIImageQualityRestorationFast(restorationBoos
tFactor);
```

PARAMETERS

Parameter	Description	Default
`restorationBoostFactor` Single, Optional	Restoration boost factor.	1.0F

RETURN VALUE

Boolean — True if restoration succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.AIImageQualityRestorationFast(1.0F);
```

12. Action.AIImageQualityRestorationFastest() Returns: Boolean

DESCRIPTION

Restores a low-quality image using the fastest AI preview model.

SYNTAX

```
let result = Action.AIImageQualityRestorationFastest();
```

RETURN VALUE

Boolean — True if restoration succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.AIImageQualityRestorationFastest();
```

13. Action.AIImageQualityRestorationSlow() Returns: Boolean

DESCRIPTION

Restores a low-quality image using the slower, more accurate AI model.

SYNTAX

```
let result = Action.AIImageQualityRestorationSlow();
```

RETURN VALUE

Boolean — True if restoration succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.AIImageQualityRestorationSlow();
```

14. Action.AIImageQualityRestorationSlowest() Returns: Boolean

DESCRIPTION

Restores a low-quality image using the highest-quality AI model.

SYNTAX

```
let result = Action.AIImageQualityRestorationSlowest();
```

RETURN VALUE

Boolean — True if restoration succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.AIImageQualityRestorationSlowest();
```

15. Action.AIImageSharpening(sharpenBoostFactor) Returns: Boolean

DESCRIPTION

Sharpens the image using AI enhancement.

SYNTAX

```
let result = Action.AIImageSharpening(sharpenBoostFactor);
```

PARAMETERS

Parameter	Description	Default
`sharpenBoostFactor` Single, Optional	AI sharpening boost factor.	2.5F

RETURN VALUE

Boolean — True if sharpening succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.AIImageSharpening(2.5F);
```

16. Action.AIImageUpscaleRealESRFast() Returns: Boolean

DESCRIPTION

Upscales the image using the fast RealESRGAN model variant.

SYNTAX

```
let result = Action.AIImageUpscaleRealESRFast();
```

RETURN VALUE

Boolean — True if upscaling succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.AIImageUpscaleRealESRFast();
```

17. Action.AIImageUpscaleRealESRGAN(upscaleMagnificationType)

Returns: Boolean

DESCRIPTION

Upscales the image using the RealESRGAN AI model.

SYNTAX

```
let result = Action.AIImageUpscaleRealESRGAN(upscaleMagnification
Type);
```

PARAMETERS

Parameter	Description	Default
`upscaleMagnificationType`	Integer representing the	4

upscaleMagnificationType Integer, Optional	Integer representing the upscaling magnification. Supported upscaleMagnificationType values: 2: 2x 4: 4x 8: 8x	4

RETURN VALUE

Boolean — True if upscaling succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.AIImageUpscaleRealESRGAN(4);
```

18. Action.AIInpainting() Returns: Boolean

DESCRIPTION

Removes unwanted image content using AI inpainting.

SYNTAX

```
let result = Action.AIInpainting();
```

RETURN VALUE

Boolean — True if inpainting succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.AIInpainting();
```

19. Action.AIMotionDeblur() Returns: Boolean

DESCRIPTION

Removes motion blur and restores sharpness in an image using deep learning.

SYNTAX

```
let result = Action.AIMotionDeblur();
```

RETURN VALUE

Boolean — True if motion deblurring succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.AIMotionDeblur();
```

20. Action.AIObjectDetection(confidenceThreshold, nmsThreshold, enableTiling, targetClasses, drawBoundingBoxes, showConfidence, boxColorRed, boxColorGreen, boxColorBlue, exportCSV, csvExportDirectory)

Returns: Boolean

DESCRIPTION

Detects objects with YOLOv3 and optionally draws boxes or exports a CSV.

SYNTAX

```
let result = Action.AIObjectDetection(confidenceThreshold, nmsThreshold, enableTiling, targetClasses, drawBoundingBoxes, showConfidence, boxColorRed, boxColorGreen, boxColorBlue, exportCSV, csvExportDirectory);
```

PARAMETERS

Parameter	Description	Default
confidenceThreshold Single, Optional	Minimum detection confidence.	0.5F
nmsThreshold Single, Optional	Non-maximum suppression threshold.	0.5F
enableTiling Boolean, Optional	Whether to process the image in tiles.	False
targetClasses String, Optional	Comma-separated target class names; empty selects all classes.	""
drawBoundingBoxes Boolean, Optional	Whether to draw detected bounding boxes.	True
showConfidence Boolean, Optional	Whether to show confidence values with boxes.	True
boxColorRed Integer, Optional	Bounding-box red component (0–255).	255
boxColorGreen Integer, Optional	Bounding-box green component (0–255).	0
boxColorBlue Integer, Optional	Bounding-box blue component (0–255).	0

exportCSV Boolean, Optional	Whether to export detection results as CSV.	False
csvExportDirectory String, Optional	Directory for CSV output.	""

RETURN VALUE

Boolean — True if detection completed; otherwise False.

EXAMPLE CODE

```
let success = Action.AIObjectDetection(0.5F, 0.4F, false, "", true, true, 255, 0, 0, false, "");
```

21. Action.AIOverExposureCorrection() Returns: Boolean

DESCRIPTION

Restores details in an overexposed image using an AI model.

SYNTAX

```
let result = Action.AIOverExposureCorrection();
```

RETURN VALUE

Boolean — True if correction succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.AIOverExposureCorrection();
```

22. Action.AISmartSelection() Returns: Boolean

DESCRIPTION

Creates an intelligent selection using image content analysis.

SYNTAX

```
let result = Action.AISmartSelection();
```

RETURN VALUE

Boolean — True if the selection was created; otherwise False.

EXAMPLE CODE

```
let success = Action.AISmartSelection();
```

23. Action.AIUnderExposureCorrection() Returns: Boolean

DESCRIPTION

Restores details in an underexposed image using an AI model.

SYNTAX

```
let result = Action.AIUnderExposureCorrection();
```

RETURN VALUE

Boolean — True if correction succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.AIUnderExposureCorrection();
```

Processing & Morphological Actions 22 functions

1. Action.AdaptiveThresholding(adaptiveBoxSize, adaptiveThresholdLevel, isObjectColorAutomatic, manualObjectColorType)

Returns: Boolean

DESCRIPTION

Applies adaptive thresholding to images with varying illumination.

SYNTAX

```
let result = Action.AdaptiveThresholding(adaptiveBoxSize, adaptiveThresholdLevel, isObjectColorAutomatic, manualObjectColorType);
```

PARAMETERS

Parameter	Description	Default
`adaptiveBoxSize` Integer, Optional	Box size used for local threshold calculation.	80
`adaptiveThresholdLevel` Integer, Optional	Threshold tuning level.	86
`isObjectColorAutomatic` Boolean, Optional	Whether to detect object color automatically.	True

manualObjectColorType Integer, Optional	Integer representing the manual object color. Supported manualObjectColorType values (used when automatic detection is disabled): 0: White 1: Black	0

RETURN VALUE

Boolean — True if thresholding succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.AdaptiveThresholding(100, 90, false, 1);
```

2. Action.BinaryOperation(logicalOperationType, firstImageSourceLocation, secondImageSourceLocation)

Returns: Boolean

DESCRIPTION

Performs a logical operation between two workspace or buffer images.

SYNTAX

```
let result = Action.BinaryOperation(logicalOperationType, firstImageSourceLocation, secondImageSourceLocation);
```

PARAMETERS

Parameter	Description	Default
logicalOperationType Integer, Optional	Integer representing the binary logical operation. Supported logicalOperationType values: 0: AND 1: OR 2: XOR	0
firstImageSourceLocation Integer, Optional	First image source index.	0
secondImageSourceLocation Integer, Optional	Second image source index.	1

RETURN VALUE

Boolean — True if the operation succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.BinaryOperation(0, 0, 1);
```

3. Action.BoundaryDetection(boundaryOverlayModeType)

Returns: Boolean

DESCRIPTION

Detects object boundaries using the selected overlay mode.

SYNTAX

```
let result = Action.BoundaryDetection(boundaryOverlayModeType);
```

PARAMETERS

Parameter	Description	Default
boundaryOverlayModeType Integer, Optional	Integer representing the boundary overlay mode. Supported boundaryOverlayModeType values: 0: WithOverlay - draw boundaries over the original image 1: WithoutOverlay - draw boundaries only	0

RETURN VALUE

Boolean — True if boundary detection succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.BoundaryDetection(0);
```

4. Action.CannyEdge(cannyHighThreshold) Returns: Boolean

DESCRIPTION

Detects edges using the Canny algorithm.

SYNTAX

```
let result = Action.CannyEdge(cannyHighThreshold);
```

PARAMETERS

Parameter	Description	Default
`cannyHighThreshold` Integer, Optional	High threshold used by the Canny detector.	100

RETURN VALUE

Boolean — True if detection succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.CannyEdge(100);
```

5. Action.ChangeNonBlackToWhite() Returns: Boolean

DESCRIPTION

Converts every non-black pixel to white.

SYNTAX

```
let result = Action.ChangeNonBlackToWhite();
```

RETURN VALUE

Boolean — True if conversion succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.ChangeNonBlackToWhite();
```

6. Action.DifferentialImage(firstImageSourceLocation, secondImageSourceLocation)

Returns: Boolean

DESCRIPTION

Computes the difference between two workspace or buffer images.

SYNTAX

```
let result = Action.DifferentialImage(firstImageSourceLocation, secondImageSourceLocation);
```

PARAMETERS

Parameter	Description	Default
`firstImageSourceLocation` Integer, Optional	First image source index.	0
`secondImageSourceLocation` Integer, Optional	Second image source index.	1

RETURN VALUE

Boolean — True if the difference image was created; otherwise False.

EXAMPLE CODE

```
let success = Action.DifferentialImage(0, 1);
```

7. Action.Dilate(morphologyKernelSize) Returns: Boolean

DESCRIPTION

Applies morphological dilation to expand foreground objects.

SYNTAX

```
let result = Action.Dilate(morphologyKernelSize);
```

PARAMETERS

Parameter	Description	Default
`morphologyKernelSize` Integer, Optional	Morphological kernel size.	3

RETURN VALUE

Boolean — True if dilation was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.Dilate(3);
```

8. Action.DilateOutline() Returns: Boolean

DESCRIPTION

Dilates traced outlines to make them thicker.

SYNTAX

```
let result = Action.DilateOutline();
```

RETURN VALUE

Boolean — True if outlines were dilated; otherwise False.

EXAMPLE CODE

```
let success = Action.DilateOutline();
```

9. Action.DrawSelectionBorder(borderColorRed, borderColorGreen, borderColorBlue, borderThicknessPixels, borderOpacityPercentage, applyToMultipleROIs)

Returns: Boolean

DESCRIPTION

Draws a colored border around the selected region or ROIs.

SYNTAX

```
let result = Action.DrawSelectionBorder(borderColorRed, borderColorGreen, borderColorBlue, borderThicknessPixels, borderOpacityPercentage, applyToMultipleROIs);
```

PARAMETERS

Parameter	Description	Default
`borderColorRed` Integer, Optional	Border red component (0–255).	0
`borderColorGreen` Integer, Optional	Border green component (0–255).	0
`borderColorBlue` Integer, Optional	Border blue component (0–255).	0
`borderThicknessPixels` Integer, Optional	Border thickness in pixels.	1
`borderOpacityPercentage` Integer, Optional	Border opacity percentage.	100
`applyToMultipleROIs` Boolean, Optional	Whether to apply to multiple ROIs.	False

RETURN VALUE

Boolean — True if the border was drawn; otherwise False.

EXAMPLE CODE

```
let success = Action.DrawSelectionBorder(0, 0, 0, 1, 100, false);
```

10. Action.EdgeDetection(edgeDetectionAlgorithmType)

Returns: Boolean

DESCRIPTION

Detects image edges using the selected algorithm.

SYNTAX

```
let result = Action.EdgeDetection(edgeDetectionAlgorithmType);
```

PARAMETERS

Parameter	Description	Default
edgeDetectionAlgorithmType Integer, Optional	Integer representing the edge detection algorithm. Supported edgeDetectionAlgorithmType values: 0: Homogeneity 1: Sobel 2: LeftHand 3: UpperLeftHand	0

RETURN VALUE

Boolean — True if detection succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.EdgeDetection(0);
```

11. Action.Erode(morphologyKernelSize) Returns: Boolean

DESCRIPTION

Applies morphological erosion to shrink foreground objects.

SYNTAX

```
let result = Action.Erode(morphologyKernelSize);
```

PARAMETERS

Parameter	Description	Default
`morphologyKernelSize` Integer, Optional	Morphological kernel size.	3

RETURN VALUE

Boolean — True if erosion was applied; otherwise False.

EXAMPLE CODE

```
let success = Action.Erode(3);
```

12. Action.ErodeOutline() Returns: Boolean

DESCRIPTION

Erodes traced outlines to make them thinner.

SYNTAX

```
let result = Action.ErodeOutline();
```

RETURN VALUE

Boolean — True if outlines were eroded; otherwise False.

EXAMPLE CODE

```
let success = Action.ErodeOutline();
```

13. Action.FillRegion(fillRegionType, fillColorRed, fillColorGreen, fillColorBlue, fillOpacityPercentage)

Returns: Boolean

DESCRIPTION

Fills the selected region or ROI with a color and opacity.

SYNTAX

```
let result = Action.FillRegion(fillRegionType, fillColorRed, fillColorGreen, fillColorBlue, fillOpacityPercentage);
```

PARAMETERS

Parameter	Description	Default
`fillRegionType` Integer, Optional	Integer representing the fill target.	0

	Supported fillRegionType values: 0: Inside - fill inside the region 1: Outside - fill outside the region	
fillColorRed Integer, Optional	Fill color red component (0–255).	0
fillColorGreen Integer, Optional	Fill color green component (0–255).	0
fillColorBlue Integer, Optional	Fill color blue component (0–255).	0
fillOpacityPercentage Integer, Optional	Fill opacity percentage.	100

RETURN VALUE

Boolean — True if the region was filled; otherwise False.

EXAMPLE CODE

```
let success = Action.FillRegion(0, 0, 0, 0, 100);
```

14. Action.HideOutlines(targetColorRed, targetColorGreen, targetColorBlue, useMostFrequentColor, repeatCount)

Returns: Boolean

DESCRIPTION

Hides object outlines using a specified or most frequent color.

SYNTAX

```
let result = Action.HideOutlines(targetColorRed, targetColorGreen, targetColorBlue, useMostFrequentColor, repeatCount);
```

PARAMETERS

Parameter	Description	Default
targetColorRed Integer, Optional	Replacement red component (0–255).	255
targetColorGreen Integer, Optional	Replacement green component (0–255).	255
targetColorBlue Integer, Optional	Replacement blue component (0–255).	255

useMostFrequentColor Boolean, Optional	Whether to use the most frequent image color instead.	True
repeatCount Integer, Optional	Number of iterations to repeat the outline hiding process (1–50).	1

RETURN VALUE

Boolean — True if outlines were hidden; otherwise False.

EXAMPLE CODE

```
let success = Action.HideOutlines(255, 255, 255, true, 3);
```

15. Action.ManualThresholding(binarizationThresholdValue)

Returns: Boolean

DESCRIPTION

Converts the image to binary using a manual intensity threshold.

SYNTAX

```
let result = Action.ManualThresholding(binarizationThresholdValue);
```

PARAMETERS

Parameter	Description	Default
binarizationThresholdValue Integer, Optional	Threshold intensity.	128

RETURN VALUE

Boolean — True if binarization succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.ManualThresholding(128);
```

16. Action.MultiStepThresholding(redChannelLevels, greenChannelLevels, blueChannelLevels)

Returns: Boolean

DESCRIPTION

Applies multilevel thresholding independently to RGB channels

Applies multilevel thresholding independently to RGB channels.

SYNTAX

```
let result = Action.MultiStepThresholding(redChannelLevels, greenChannelLevels, blueChannelLevels);
```

PARAMETERS

Parameter	Description	Default
`redChannelLevels` Integer, Optional	Number of red-channel intensity levels.	5
`greenChannelLevels` Integer, Optional	Number of green-channel intensity levels.	5
`blueChannelLevels` Integer, Optional	Number of blue-channel intensity levels.	5

RETURN VALUE

Boolean — True if thresholding succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.MultiStepThresholding(5, 5, 5);
```

17. Action.Outline(isInverted, applyBinarization) Returns: Boolean

DESCRIPTION

Traces object outlines while always preserving the original pixel colors on the extracted outlines.

SYNTAX

```
let result = Action.Outline(isInverted, applyBinarization);
```

PARAMETERS

Parameter	Description	Default
`isInverted` Boolean, Optional	Whether to invert the binarization or background result before extracting outlines (dark objects on a light background).	False
`applyBinarization` Boolean, Optional	Whether to apply threshold-based pre-binarization for boundary detection. Set to False to skip binarization and extract boundaries	True

	binarization and extract boundaries directly from all non-zero color regions (e.g., after Region Coloring) without brightness threshold loss. Original outline colors are preserved in both modes.

RETURN VALUE

Boolean — True if outlines were traced; otherwise False.

EXAMPLE CODE

```
let success = Action.Outline(false, false);
```

18. Action.RegionColoring(connectivity) Returns: Boolean

DESCRIPTION

Colors connected image regions to distinguish them.

SYNTAX

```
let result = Action.RegionColoring(connectivity);
```

PARAMETERS

Parameter	Description	Default
connectivity Integer, Optional	Integer representing the pixel connectivity. Supported connectivity values: 4: 4-connected (FourWay) 8: 8-connected (EightWay); any other value is treated as 8	8

RETURN VALUE

Boolean — True if the regions were colored; otherwise False.

EXAMPLE CODE

```
let success = Action.RegionColoring(8);
```

19. Action.RemoveDots(dotMode, maxDotSize, connectivity)

Returns: Boolean

DESCRIPTION

Removes small connected components from a binary image.

SYNTAX

```
let result = Action.RemoveDots(dotMode, maxDotSize, connectivit
y);
```

PARAMETERS

Parameter	Description	Default
`dotMode` Integer	Integer representing the target dots. Supported dotMode values: 0: Black dots 1: White dots (also used for any other value) 2: Both black and white dots	–
`maxDotSize` Integer	Maximum component area in pixels to remove.	–
`connectivity` Integer, Optional	Integer representing pixel connectivity. Supported connectivity values: 4: 4-connected (FourWay) 8: 8-connected (EightWay); any other value is treated as 8	8

RETURN VALUE

Boolean — True if dot removal succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.RemoveDots(2, 20, 8);
```

20. Action.Skeletonize(isExtended, isInverted, minBranchLength, tailLength)

Returns: Boolean

DESCRIPTION

Reduces binary objects to one-pixel-wide skeletons.

SYNTAX

```
let result = Action.Skeletonize(isExtended, isInverted, minBranch
Length, tailLength);
```

PARAMETERS

Parameter	Description	Default

isExtended Boolean, Optional	Whether to use extended skeleton processing.	True
isInverted Boolean, Optional	Whether to invert the binary image before processing.	False
minBranchLength Integer, Optional	Minimum branch length to retain.	15
tailLength Integer, Optional	Tail length setting for skeleton processing.	10

RETURN VALUE

Boolean — True if skeletonization succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.Skeletonize(true, false, 15, 10);
```

21. Action.SuperPixelSegmentation(numSuperpixels, compactness, fillWithWhite)

Returns: Boolean

DESCRIPTION

Segments the image into superpixels.

SYNTAX

```
let result = Action.SuperPixelSegmentation(numSuperpixels, compactness, fillWithWhite);
```

PARAMETERS

Parameter	Description	Default
numSuperpixels Integer, Optional	Requested number of superpixels.	400
compactness Single, Optional	Superpixel compactness factor.	30.0F
fillWithWhite Boolean, Optional	Whether to fill superpixel regions with white.	False

RETURN VALUE

Boolean — True if segmentation succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.SuperPixelSegmentation(400, 30.0F, false);
```

22. Action.WatershedSegmentation(minDistanceThreshold, smoothingRadius, boundaryThickness, boundaryColorRed, boundaryColorGreen, boundaryColorBlue)

Returns: Boolean

DESCRIPTION

Segments touching objects using watershed separation.

SYNTAX

```
let result = Action.WatershedSegmentation(minDistanceThreshold, smoothingRadius, boundaryThickness, boundaryColorRed, boundaryColorGreen, boundaryColorBlue);
```

PARAMETERS

Parameter	Description	Default
`minDistanceThreshold` Integer, Optional	Minimum distance threshold between object centers.	10
`smoothingRadius` Integer, Optional	Pre-segmentation smoothing radius.	3
`boundaryThickness` Integer, Optional	Boundary thickness in pixels.	1
`boundaryColorRed` Integer, Optional	Boundary red component (0–255).	255
`boundaryColorGreen` Integer, Optional	Boundary green component (0–255).	0
`boundaryColorBlue` Integer, Optional	Boundary blue component (0–255).	0

RETURN VALUE

Boolean — True if segmentation succeeded; otherwise False.

EXAMPLE CODE

```
let success = Action.WatershedSegmentation(10, 3, 1, 255, 0, 0);
```

System & Workspace Actions 8 functions

1. Action.ActivateUndoEngine() Returns: Void

DESCRIPTION

Activates undo tracking for workspace operations.

SYNTAX

```
Action.ActivateUndoEngine();
```

RETURN VALUE

`Void` — Void.

EXAMPLE CODE

```
Action.ActivateUndoEngine();
```

2. Action.DeactivateUndoEngine() Returns: Void

DESCRIPTION

Deactivates undo tracking for workspace operations.

SYNTAX

```
Action.DeactivateUndoEngine();
```

RETURN VALUE

`Void` — Void.

EXAMPLE CODE

```
Action.DeactivateUndoEngine();
```

3. Action.GetImage(sourceLocation) Returns: Bitmap

DESCRIPTION

Returns a clone of the workspace image or an image-buffer image.

SYNTAX

```
let result = Action.GetImage(sourceLocation);
```

PARAMETERS

Parameter	Description	Default
sourceLocation Integer, Optional	0 selects the workspace; 1–10 selects a buffer.	0

RETURN VALUE

Bitmap — Bitmap clone, or Nothing when unavailable or export is not allowed.

EXAMPLE CODE

```
let bitmap = Action.GetImage(0);
```

4. Action.SaveImage(sourceBitmap, destFileName, jpegQuality)

Returns: String

DESCRIPTION

Saves a bitmap image to the specified file path.

SYNTAX

```
let result = Action.SaveImage(sourceBitmap, destFileName, jpegQua
lity);
```

PARAMETERS

Parameter	Description	Default
sourceBitmap Bitmap	Bitmap to save.	–
destFileName String	Destination file path.	–
jpegQuality Integer, Optional	JPEG quality from 0 to 100.	90

RETURN VALUE

String — String containing the save result or an error message.

EXAMPLE CODE

```
let message = Action.SaveImage(bitmap, "C:₩Images₩result.jpg", 90);
```

5. Action.SaveImageDialog(targetBitmap, selectedExtension, selectedQuality)

Returns: Void

DESCRIPTION

Opens a dialog to save an image with the selected format and quality.

SYNTAX

```
Action.SaveImageDialog(targetBitmap, selectedExtension, selectedQuality);
```

PARAMETERS

Parameter	Description	Default
targetBitmap Bitmap	Bitmap to save.	–
selectedExtension String	Selected file extension.	–
selectedQuality Integer	JPEG quality setting.	–

RETURN VALUE

`Void` — Void. Throws an exception if export is not allowed.

EXAMPLE CODE

```
Action.SaveImageDialog(bitmap, ".jpg", 90);
```

6. Action.SaveWorkspaceAs() Returns: Void

DESCRIPTION

Saves the current workspace with a new name.

SYNTAX

```
Action.SaveWorkspaceAs();
```

RETURN VALUE

`Void` — Void. Throws an exception if export is not allowed.

EXAMPLE CODE

```
Action.SaveWorkspaceAs();
```

7. Action.SaveWorkspaceSelectionAs() Returns: Void

DESCRIPTION

Saves the current workspace selection as a new file.

SYNTAX

```
Action.SaveWorkspaceSelectionAs();
```

RETURN VALUE

`Void` — Void. Throws an exception if export is not allowed.

EXAMPLE CODE

```
Action.SaveWorkspaceSelectionAs();
```

8. Action.SetImage(newImage, targetLocation) Returns: Boolean

DESCRIPTION

Sets the workspace image or an image-buffer image from a bitmap clone.

SYNTAX

```
let result = Action.SetImage(newImage, targetLocation);
```

PARAMETERS

Parameter	Description	Default
`newImage` Bitmap	Image to assign.	–
`targetLocation` Integer, Optional	0 selects the workspace; 1–10 selects a buffer.	0

RETURN VALUE

`Boolean` — True when the image is assigned; otherwise False.

EXAMPLE CODE

```
let success = Action.SetImage(bitmap, 0);
```

2. DNNClassifier (ScriptDNNClassifierHelper) 3 Functions

> **Class Overview:** Provides DNN-based image classification capabilities. Allows scripts to dynamically load trained ML.NET models and predict class labels for images.

Neural & AI Processing Actions 3 functions

1. DNNClassifier.GetLastError() Returns: String

DESCRIPTION

Returns the most recent error message produced by the helper.

SYNTAX

```
let result = DNNClassifier.GetLastError();
```

RETURN VALUE

String — The last error message, or an empty string when no error is recorded.

EXAMPLE CODE

```
let errorMessage = DNNClassifier.GetLastError();
```

2. DNNClassifier.LoadModel(modelFilename, forceReload)

Returns: Boolean

DESCRIPTION

Loads an ML.NET image-classification model from the Classification Models folder.

SYNTAX

```
let result = DNNClassifier.LoadModel(modelFilename, forceReload);
```

PARAMETERS

Parameter	Description	Default
`modelFilename` String	Model ZIP file name.	–
`forceReload` Boolean, Optional	Whether to reload the model when the same model is already loaded.	False

RETURN VALUE

Boolean — True when the model is loaded or already loaded; otherwise False.

EXAMPLE CODE

```
let loaded = DNNClassifier.LoadModel("model.zip", true);
```

3. DNNClassifier.Predict(img) Returns: String

DESCRIPTION

Predicts the class label for the supplied bitmap image.

SYNTAX

```
let result = DNNClassifier.Predict(img);
```

PARAMETERS

Parameter	Description	Default
`img` Bitmap	Image to classify.	–

RETURN VALUE

String — Predicted class label, or an empty string if prediction fails.

EXAMPLE CODE

```
let label = DNNClassifier.Predict(bitmap);
```

3. FaceDetector (ScriptFaceDetectorHelper) 12 Functions

Neural & AI Processing Actions 12 functions

1. FaceDetector.ApplyFaceMaskToWorkspace(useRotatedEllipse, scaleExpand)

Returns: Boolean

DESCRIPTION

Applies the detected face regions directly to the active workspace selection mask (AdvSel).

SYNTAX

```
let result = FaceDetector.ApplyFaceMaskToWorkspace(useRotatedEllipse, scaleExpand);
```

PARAMETERS

Parameter	Description	Default
`useRotatedEllipse` Boolean, Optional	True to use rotated elliptical masks; False for rectangular masks.	True
`scaleExpand` Double, Optional	Expansion scale factor for face ROI coverage.	1.08

RETURN VALUE

Boolean — True if the face mask was successfully applied to the workspace selection; otherwise False.

EXAMPLE CODE

```
let ok = FaceDetector.ApplyFaceMaskToWorkspace(true, 1.08);
```

2. FaceDetector.Detect(img, confThreshold, nmsThreshold, useTiling)

Returns: Integer

DESCRIPTION

Detects faces and 6-point facial landmarks in an image and stores the detection results.

SYNTAX

```
let result = FaceDetector.Detect(img, confThreshold, nmsThreshold, useTiling);
```

PARAMETERS

Parameter	Description	Default
`img` Bitmap	Image to analyze.	–
`confThreshold` Double, Optional	Minimum confidence threshold.	0.45
`nmsThreshold` Double, Optional	Non-maximum suppression threshold.	0.4

useTiling Boolean, Optional	Whether to use multi-scale tiled inference for high-res/group images.	True

RETURN VALUE

Integer — Number of detected faces, or 0 if detection fails.

EXAMPLE CODE

```
let count = FaceDetector.Detect(bitmap, 0.45, 0.4, true);
```

3. FaceDetector.GetAllResults()

Returns: ONNX_FaceDetection.FaceDetectionResult()

DESCRIPTION

Returns all face detection results from the most recent detection.

SYNTAX

```
let result = FaceDetector.GetAllResults();
```

RETURN VALUE

ONNX_FaceDetection.FaceDetectionResult() — Array of face detection results, or an empty array when no results are available.

EXAMPLE CODE

```
let results = FaceDetector.GetAllResults();
```

4. FaceDetector.GetFaceCount() Returns: Integer

DESCRIPTION

Returns the total number of faces from the most recent detection.

SYNTAX

```
let result = FaceDetector.GetFaceCount();
```

RETURN VALUE

Integer — Number of detected faces.

EXAMPLE CODE

```
let count = FaceDetector.GetFaceCount();
```

5. FaceDetector.GetFaceKeypoints(faceIndex) Returns: PointF()

DESCRIPTION

Returns the 6 facial keypoints (0:LeftEye, 1:RightEye, 2:NoseTip, 3:MouthCenter, 4:LeftEar, 5:RightEar) for a specific 0-based face index.

SYNTAX

```
let result = FaceDetector.GetFaceKeypoints(faceIndex);
```

PARAMETERS

Parameter	Description	Default
`faceIndex` Integer	Zero-based index of the detected face.	–

RETURN VALUE

PointF() — Array of 6 PointF coordinates, or an empty array if the index is out of range.

EXAMPLE CODE

```
let kps = FaceDetector.GetFaceKeypoints(0);
```

6. FaceDetector.GetFaceLocations() Returns: RectangleF()

DESCRIPTION

Returns bounding rectangles for all detected faces.

SYNTAX

```
let result = FaceDetector.GetFaceLocations();
```

RETURN VALUE

RectangleF() — Array of bounding rectangles (RectangleF) for all detected faces.

EXAMPLE CODE

```
let locations = FaceDetector.GetFaceLocations();
```

7. FaceDetector.GetFaceMaskImage(img, useRotatedEllipse, scaleExpand)

Returns: Bitmap

DESCRIPTION

Generates a strict binary mask bitmap (White = Detected Faces, Black = Background) from the most recent detection results.

SYNTAX

```
let result = FaceDetector.GetFaceMaskImage(img, useRotatedEllipse, scaleExpand);
```

PARAMETERS

Parameter	Description	Default
img Bitmap	Source image used to match dimensions and resolution.	–
useRotatedEllipse Boolean, Optional	True to draw rotated ellipses aligned with eye tilt; False for rectangles.	True
scaleExpand Double, Optional	Expansion scale factor for face ROI coverage.	1.08

RETURN VALUE

Bitmap — Binary mask bitmap (24bppRgb), or Nothing if generation fails.

EXAMPLE CODE

```
let maskBmp = FaceDetector.GetFaceMaskImage(bitmap, true, 1.08);
```

8. FaceDetector.GetFaceRollAngle(faceIndex) Returns: Single

DESCRIPTION

Returns the head tilt (roll) angle in degrees for a specific 0-based face index.

SYNTAX

```
let result = FaceDetector.GetFaceRollAngle(faceIndex);
```

PARAMETERS

Parameter	Description	Default
faceIndex Integer	Zero-based index of the detected face.	–

RETURN VALUE

Single — Head tilt angle in degrees, or 0.0 if the index is out of range.

EXAMPLE CODE

```
let tilt = FaceDetector.GetFaceRollAngle(0);
```

9. FaceDetector.GetLastError() Returns: String

DESCRIPTION

Returns the most recent error message produced by the helper.

SYNTAX

```
let result = FaceDetector.GetLastError();
```

RETURN VALUE

String — The last error message, or an empty string when no error is recorded.

EXAMPLE CODE

```
let errorMessage = FaceDetector.GetLastError();
```

10. FaceDetector.GetRenderedImage(img, showBoundingBox, showFaceId, showConfidence, showKeypoints, useRotatedEllipse, colorName)

Returns: Bitmap

DESCRIPTION

Returns a copy of the image rendered with independent options for face ROI, ID, confidence, and 6 keypoints.

SYNTAX

```
let result = FaceDetector.GetRenderedImage(img, showBoundingBox, showFaceId, showConfidence, showKeypoints, useRotatedEllipse, colorName);
```

PARAMETERS

Parameter	Description	Default
img Bitmap	Source image to render.	–
showBoundingBox Boolean, Optional	Whether to draw the bounding ROI shape.	True
showFaceId	Whether to display the face ID	True

Boolean, Optional	number (#1, #2...).	
showConfidence Boolean, Optional	Whether to display confidence values.	True
showKeypoints Boolean, Optional	Whether to draw the 6 facial keypoints.	True
useRotatedEllipse Boolean, Optional	True to draw rotated ellipses; False for rectangles.	True
colorName String, Optional	Known .NET color name for ROI boxes.	"Red"

RETURN VALUE

Bitmap — Rendered bitmap, or Nothing if rendering fails.

EXAMPLE CODE

```
let rendered = FaceDetector.GetRenderedImage(bitmap, true, true,
true, true, true, "Red");
```

11. FaceDetector.HasFaces() Returns: Boolean

DESCRIPTION

Checks whether at least one face was detected in the most recent analysis.

SYNTAX

```
let result = FaceDetector.HasFaces();
```

RETURN VALUE

Boolean — True if one or more faces were detected; otherwise False.

EXAMPLE CODE

```
let found = FaceDetector.HasFaces();
```

12. FaceDetector.Initialize() Returns: Boolean

DESCRIPTION

Initializes the BlazeFace detector with its default model and settings.

SYNTAX

```
let result = FaceDetector.Initialize();
```

RETURN VALUE

Boolean — True when the detector is initialized or already available; otherwise False.

EXAMPLE CODE

```
let ready = FaceDetector.Initialize();
```

4. ImgProc (ScriptImgProcHelper) 271 Functions

Color & Channel Actions 69 functions

1. ImgProc.ARGBToARGBInteger(_AlphaValue, _RedValue, _GreenValue, _BlueValue)

Returns: Integer

DESCRIPTION

Converts ARGB values to integer.

SYNTAX

```
let result = ImgProc.ARGBToARGBInteger(_AlphaValue, _RedValue, _GreenValue, _BlueValue);
```

PARAMETERS

Parameter	Description	Default
`_AlphaValue` Integer	Alpha channel value (0-255).	–
`_RedValue` Integer	Red channel value (0-255).	–
`_GreenValue` Integer	Green channel value (0-255).	–
`_BlueValue` Integer	Blue channel value (0-255).	–

RETURN VALUE

Integer — ARGB color as integer, or 0 if conversion fails

Integer — ARGB color as integer, or 0 if conversion fails.

EXAMPLE CODE

```
let color = ImgProc.ARGBToARGBInteger(255, 255, 0, 0);
```

2. ImgProc.ARGBToColor(AlphaInByte, RedInByte, GreenInByte, BlueInByte)

Returns: Color

DESCRIPTION

Converts ARGB bytes to Color object.

SYNTAX

```
let result = ImgProc.ARGBToColor(AlphaInByte, RedInByte, GreenInByte, BlueInByte);
```

PARAMETERS

Parameter	Description	Default
AlphaInByte Integer	Alpha channel value (0-255).	–
RedInByte Integer	Red channel value (0-255).	–
GreenInByte Integer	Green channel value (0-255).	–
BlueInByte Integer	Blue channel value (0-255).	–

RETURN VALUE

Color — Color object, or default color if conversion fails.

EXAMPLE CODE

```
let clr = ImgProc.ARGBToColor(255, 255, 0, 0);
```

3. ImgProc.Change_Color(SourceImage, ColorOfInterest_RGBInt, ChangeToR, ChangeToG, ChangeToB, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Changes specified color to new color throughout image.

SYNTAX

```
let result = ImgProc.Change_Color(SourceImage, ColorOfInterest_RGBInt, ChangeToR, ChangeToG, ChangeToB, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to modify. Type: Bitmap; required; passed by reference.	–
ColorOfInterest_RGBInt Integer	RGB color to replace.	–
ChangeToR Byte	New red channel value.	–
ChangeToG Byte	New green channel value.	–
ChangeToB Byte	New blue channel value.	–
IsShowProgress Boolean, Optional	Whether to show progress.	`True`
MaskImage Bitmap, Optional	Optional mask to constrain operation.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Bitmap` — Modified image, or Nothing if operation fails.

EXAMPLE CODE

```
let modified = ImgProc.Change_Color(bitmap, RGB(255,0,0), 0, 255, 0);
```

4. ImgProc.Change_Color_ByRGB(SourceImage, ColorOfInterestR, ColorOfInterestG, ColorOfInterestB, ChangeToR, ChangeToG, ChangeToB, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Changes specified RGB color to new RGB color.

SYNTAX

```
let result = ImgProc.Change_Color_ByRGB(SourceImage, ColorOfInterestR, ColorOfInterestG, ColorOfInterestB, ChangeToR, ChangeToG, ChangeToB, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to modify. Type: Bitmap; required; passed by reference.	–
ColorOfInterestR Byte	Red channel of color to replace.	–
ColorOfInterestG Byte	Green channel of color to replace.	–
ColorOfInterestB Byte	Blue channel of color to replace.	–
ChangeToR Byte	New red channel value.	–
ChangeToG Byte	New green channel value.	–
ChangeToB Byte	New blue channel value.	–
IsShowProgress Boolean, Optional	Whether to show progress.	`True`
MaskImage Bitmap, Optional	Optional mask to constrain operation.	`Nothing`

MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Modified image, or Nothing if operation fails.

EXAMPLE CODE

```
let modified = ImgProc.Change_Color_ByRGB(bitmap, 255, 0, 0, 0, 255, 0);
```

5. ImgProc.Check_RGBInTextBox(Text_R, Text_G, Text_B)

Returns: Void

DESCRIPTION

Validates RGB values in text boxes.

SYNTAX

```
ImgProc.Check_RGBInTextBox(Text_R, Text_G, Text_B);
```

PARAMETERS

Parameter	Description	Default
Text_R TextBox	TextBox containing red value.	–
Text_G TextBox	TextBox containing green value.	–
Text_B TextBox	TextBox containing blue value.	–

RETURN VALUE

Void — None.

EXAMPLE CODE

```
ImgProc.Check_RGBInTextBox(txtRed, txtGreen, txtBlue);
```

6. ImgProc.CIELab(original, lShift, aShift, bShift, MaskImage,

MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Converts image to CIELab color space with adjustable L, a, b channels.

SYNTAX

```
let result = ImgProc.CIELab(original, lShift, aShift, bShift, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
original Bitmap	Source image to convert.	–
lShift Integer	Lightness shift (-127 to 127).	–
aShift Integer	a channel shift green-to-red (-127 to 127).	–
bShift Integer	b channel shift blue-to-yellow (-127 to 127).	–
MaskImage Image, Optional	Optional mask to constrain conversion.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — CIELab-converted image, or Nothing if conversion fails.

EXAMPLE CODE

```
let cielab = ImgProc.CIELab(bitmap, 0, 10, -10);
```

7. ImgProc.ColorDeconvolution(SourceImage, StainType, ShowColor, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Applies color deconvolution to separate stains in histological images using a high-speed parallel algorithm.

SYNTAX

```
let result = ImgProc.ColorDeconvolution(SourceImage, StainType, ShowColor, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
`SourceImage` Bitmap	Source image to process.	–
`StainType` Integer	Integer representing the stain type. Supported values: 0: H&E - Hematoxylin (Nuclei) 1: H&E - Eosin (Cytoplasm) 2: H-DAB - Hematoxylin (Background) 3: H-DAB - DAB (Protein Marker) 4: Masson's Trichrome - Methyl Blue (Collagen) 5: Masson's Trichrome - Ponceau Fuchsin (Muscle) 6: H-AEC - Hematoxylin (Nuclei) 7: H-AEC - AEC (Red Marker) 8: Methyl Green-DAB - Methyl Green (Nuclei) 9: Methyl Green-DAB - DAB (Protein Marker) 10: H-PAS - Hematoxylin (Nuclei) 11: H-PAS - PAS (Glycogen / Basement Membrane) 12: Alcian Blue & H - Alcian Blue (Acid Mucins / GAG) 13: Alcian Blue & H - Hematoxylin (Nuclei) 14: Giemsa - Methylene Blue (Nuclei) 15: Giemsa - Eosin	–

	(Erythrocytes / Cytoplasm) 16: Feulgen-Light Green - Feulgen (DNA) 17: Feulgen-Light Green - Light Green (Cytoplasm) 18: Safranin-O & Fast Green - Safranin-O (Lignin / Cartilage) 19: Safranin-O & Fast Green - Fast Green (Cellulose / Cytoplasm) 20: H&E-DAB (Triple) - Hematoxylin (Nuclei) 21: H&E-DAB (Triple) - Eosin (Cytoplasm) 22: H&E-DAB (Triple) - DAB (Protein Marker)	
ShowColor Boolean	Whether to apply pseudo-coloring to the extracted stain result.	–
IsShowProgress Boolean, Optional	Whether to show processing progress.	`True`
MaskImage Bitmap, Optional	Optional mask to constrain operation.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Bitmap` — Deconvoluted image, or Nothing if operation fails.

EXAMPLE CODE

```
let deconvoluted = ImgProc.ColorDeconvolution(bitmap, 0, true);
```

8. ImgProc.ColorHexToColorInteger(ColorHex) Returns: Integer

DESCRIPTION

Converts hexadecimal color string to integer.

SYNTAX

```
let result = ImgProc.ColorHexToColorInteger(ColorHex);
```

PARAMETERS

Parameter	Description	Default
ColorHex String	Hexadecimal color string (e.g., "#FF0000"). Type: String; required; passed by reference.	–

RETURN VALUE

Integer — Integer color value, or 0 if conversion fails.

EXAMPLE CODE

```
let colorInt = ImgProc.ColorHexToColorInteger("#FF0000");
```

9. ImgProc.ColorHexToRGB(HexColorStr, R, G, B) Returns: Void

DESCRIPTION

Converts hexadecimal color string to RGB components.

SYNTAX

```
ImgProc.ColorHexToRGB(HexColorStr, R, G, B);
```

PARAMETERS

Parameter	Description	Default
HexColorStr String	Hexadecimal color string (e.g., "#FF0000").	–
R Byte	Output red component. Type: Byte; required; passed by reference.	–
G Byte	Output green component. Type: Byte; required; passed by reference.	–
B Byte	Output blue component. Type: Byte; required; passed by reference.	–

RETURN VALUE

Void

EXAMPLE CODE

```
ImgProc.ColorHexToRGB("#FF0000", r, g, b);
```

10. ImgProc.ColorIntegerToARGB(PixelValue) Returns: Void

DESCRIPTION

Displays ARGB values of a color integer to console output.

SYNTAX

```
ImgProc.ColorIntegerToARGB(PixelValue);
```

PARAMETERS

Parameter	Description	Default
PixelValue Integer	Color value as integer.	–

RETURN VALUE

`Void` — Void; prints ARGB components to console.

EXAMPLE CODE

```
ImgProc.ColorIntegerToARGB(4294967295);
```

11. ImgProc.ColorIntegerToRGB(PixelValue) Returns: Void

DESCRIPTION

Displays RGB values of a color integer to console output.

SYNTAX

```
ImgProc.ColorIntegerToRGB(PixelValue);
```

PARAMETERS

Parameter	Description	Default
PixelValue Integer	Color value as integer.	–

RETURN VALUE

`Void` — Void; prints RGB components to console.

EXAMPLE CODE

```
ImgProc.ColorIntegerToRGB(255);
```

12. ImgProc.ColorIntegerToRGBGHSI(ColorValue, RedValue, GreenValue, BlueValue, GrayValue, HueValue, SaturationValue, IntensityValue)

Returns: Void

DESCRIPTION

Decomposes color to ARGB and HSI components.

SYNTAX

```
ImgProc.ColorIntegerToRGBGHSI(ColorValue, RedValue, GreenValue, BlueValue, GrayValue, HueValue, SaturationValue, IntensityValue);
```

PARAMETERS

Parameter	Description	Default
ColorValue Integer	Integer color value to decompose.	–
RedValue Byte	Output red component. Type: Byte; required; passed by reference.	–
GreenValue Byte	Output green component. Type: Byte; required; passed by reference.	–
BlueValue Byte	Output blue component. Type: Byte; required; passed by reference.	–
GrayValue Byte	Output gray component. Type: Byte; required; passed by reference.	–
HueValue Single	Output hue component. Type: Single; required; passed by reference.	–
SaturationValue Single	Output saturation component. Type: Single; required; passed by reference.	–
IntensityValue Single	Output intensity component. Type: Single; required; passed by reference.	–

RETURN VALUE

Void

EXAMPLE CODE

```
ImgProc.ColorIntegerToRGBGHSI(color, r, g, b, gray, hue, sat, int);
```

13. ImgProc.ColorRangeSelection(SourceImage, targetR, targetG, targetB, colorSpace, isFastAlgorithm, tolerance, maintainOrgColor, isShowProgress, maskImage, maskRegionBoundary, isMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Selects or isolates pixels matching the specified target RGB color within a given tolerance.

SYNTAX

```
let result = ImgProc.ColorRangeSelection(SourceImage, targetR, targetG, targetB, colorSpace, isFastAlgorithm, tolerance, maintainOrgColor, isShowProgress, maskImage, maskRegionBoundary, isMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to process.	–
targetR Integer	Target color red component (0-255).	–
targetG Integer	Target color green component (0-255).	–
targetB Integer	Target color blue component (0-255).	–
colorSpace String, Optional	Color space model ('RGB', 'HSI', 'Lab', or 'Chroma').	"RGB"
isFastAlgorithm Boolean, Optional	Whether to use Fast (Manhattan) distance instead of Standard (Euclidean).	False

	Ignored when colorSpace is 'Chroma'.	
tolerance Integer, Optional	Color matching tolerance threshold (0-255).	40
maintainOrgColor Boolean, Optional	Whether to preserve original pixel colors for matched areas (False converts matched areas to white).	True
isShowProgress Boolean, Optional	Whether to report progress events.	False
maskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
maskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
isMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

`Bitmap` — Processed Bitmap with color range selection applied, or Nothing if processing fails.

EXAMPLE CODE

```
let selected = ImgProc.ColorRangeSelection(bitmap, 0, 255, 0,
"Chroma", false, 40, true);
```

14. ImgProc.ColorToColorHEX(SourceColor) Returns: String

DESCRIPTION

Converts RGB Color object to hexadecimal string representation.

SYNTAX

```
let result = ImgProc.ColorToColorHEX(SourceColor);
```

PARAMETERS

Parameter	Description	Default
SourceColor Color	Color object to convert.	–

RETURN VALUE

`String` — Hexadecimal color string (e.g., "#FF0000"), or empty string if conversion fails.

EXAMPLE CODE

```
let hex = ImgProc.ColorToColorHEX(Color.Red);
```

15. ImgProc.Convert_Gray24Bit_To_8bppIndexed(srcBitmap)

Returns: Bitmap

DESCRIPTION

Converts 24-bit grayscale to 8-bit indexed image format.

SYNTAX

```
let result = ImgProc.Convert_Gray24Bit_To_8bppIndexed(srcBitmap);
```

PARAMETERS

Parameter	Description	Default
srcBitmap Bitmap	Source 24-bit grayscale bitmap.	–

RETURN VALUE

`Bitmap` — 8-bit indexed image, or Nothing if conversion fails.

EXAMPLE CODE

```
let indexed = ImgProc.Convert_Gray24Bit_To_8bppIndexed(bitmap);
```

16. ImgProc.Convert_To_rgChromaticity(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Converts RGB to rg chromaticity coordinates.

SYNTAX

```
let result = ImgProc.Convert_To_rgChromaticity(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to convert. Type: Bitmap; required; passed by reference.	–
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — rg chromaticity image, or Nothing if conversion fails.

EXAMPLE CODE

```
let rgChromaticity = ImgProc.Convert_To_rgChromaticity(bitmap);
```

17. ImgProc.Convert_To_rgChromaticity_Parallel(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Converts RGB to rg chromaticity with parallel processing.

SYNTAX

```
let result = ImgProc.Convert_To_rgChromaticity_Parallel(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to convert. Type: Bitmap; required; passed by reference.	–
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing

IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — rg chromaticity image, or Nothing if conversion fails.

EXAMPLE CODE

```
let rgChromaticity =
ImgProc.Convert_To_rgChromaticity_Parallel(bitmap);
```

18. ImgProc.Delete_BlueChannel(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Removes blue channel from image.

SYNTAX

```
let result = ImgProc.Delete_BlueChannel(SourceImage, MaskImage, M
askRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to process.	–
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Image with blue channel removed, or Nothing if operation fails.

EXAMPLE CODE

```
let noBlue = ImgProc.Delete_BlueChannel(bitmap);
```

19. ImgProc.Delete_GreenChannel(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Removes green channel from image.

SYNTAX

```
let result = ImgProc.Delete_GreenChannel(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to process.	–
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Image with green channel removed, or Nothing if operation fails.

EXAMPLE CODE

```
let noGreen = ImgProc.Delete_GreenChannel(bitmap);
```

20. ImgProc.Delete_RedChannel(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Removes red channel from image.

SYNTAX

```
let result = ImgProc.Delete_RedChannel(SourceImage, MaskImage, Ma
```

```
skRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to process.	–
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Image with red channel removed, or Nothing if operation fails.

EXAMPLE CODE

```
let noRed = ImgProc.Delete_RedChannel(bitmap);
```

21. ImgProc.Digitalize(ColorChannelValue, LevelCount) Returns: Byte

DESCRIPTION

Quantizes color channel value to discrete levels.

SYNTAX

```
let result = ImgProc.Digitalize(ColorChannelValue, LevelCount);
```

PARAMETERS

Parameter	Description	Default
ColorChannelValue Single	Color value to quantize.	–
LevelCount Integer	Number of discrete levels.	–

RETURN VALUE

Byte — Quantized color value.

EXAMPLE CODE

```
let quantized = ImgProc.Digitalize(128.5, 8);
```

22. ImgProc.GrayFrom_Blue(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Extracts blue channel as grayscale image.

SYNTAX

```
let result = ImgProc.GrayFrom_Blue(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source color image. Type: Bitmap; required; passed by reference.	–
MaskImage Bitmap, Optional	Optional mask to constrain extraction.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Blue channel as grayscale, or Nothing if extraction fails.

EXAMPLE CODE

```
let grayBlue = ImgProc.GrayFrom_Blue(bitmap);
```

23. ImgProc.GrayFrom_Green(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Extracts green channel as grayscale.

SYNTAX

```
let result = ImgProc.GrayFrom_Green(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source color image. Type: Bitmap; required; passed by reference.	–
MaskImage Bitmap, Optional	Optional mask to constrain extraction.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Green channel as grayscale, or Nothing if extraction fails.

EXAMPLE CODE

```
let grayGreen = ImgProc.GrayFrom_Green(bitmap);
```

24. ImgProc.GrayFrom_Red(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Extracts red channel as grayscale image.

SYNTAX

```
let result = ImgProc.GrayFrom_Red(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source color image. Type: Bitmap; required; passed	–

	by reference.	
MaskImage Bitmap, Optional	Optional mask to constrain extraction.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Red channel as grayscale, or Nothing if extraction fails.

EXAMPLE CODE

```
let grayRed = ImgProc.GrayFrom_Red(bitmap);
```

25. ImgProc.GrayScaling(SourceImage, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Converts colored image to grayscale using luminosity method.

SYNTAX

```
let result = ImgProc.GrayScaling(SourceImage, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to convert.	–
IsShowProgress Boolean, Optional	Whether to display progress.	True
MaskImage Bitmap, Optional	Optional mask to constrain conversion.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

`Bitmap` — Grayscale image, or Nothing if conversion fails.

EXAMPLE CODE

```
let gray = ImgProc.GrayScaling(bitmap, true);
```

26. ImgProc.GrayToColor(SourceImage, MapType, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Applies color map to grayscale image for visualization.

SYNTAX

```
let result = ImgProc.GrayToColor(SourceImage, MapType, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source grayscale image.	–
MapType ImgProcessing.ColorMapType	Color map type to apply. Type: ImgProcessing.ColorMapType; required. Supported values: 0: RedBlue 1: Viridis 2: Parula 3: Plasma 4: Inferno 5: Magma 6: Cividis 7: Jet 8: Turbo 9: Rainbow 10: Spectral 11: HSV 12: Ocean 13: Seismic 14: RdBu 15: Coolwarm	–

	16: bwr 17: BrBG 18: PiYG 19: Hot 20: Cool 21: Spring 22: Summer 23: Autumn 24: Winter 25: Bone 26: Copper 27: Pink 28: Terrain 29: Twilight 30: Rocket (Black - Blue - Red - Light Yellow) 31: Mako (Black - Deep Blue - Purple - Light Pink) 32: Cubehelix (Astronomy standard, brightness preserved in grayscale) 33: YlGnBu (Yellow - Green - Blue) 34: RdYlGn (Red - Yellow - Green) 35: PuOr (Purple - White - Orange) 36: Ironbow (Thermal: Black - Purple - Red - Yellow - White) 37: WhiteHot (Thermal: Cold Black - Hot White) 38: BlackHot (Thermal: Cold White - Hot Black) 39: Lava (Black - Dark Red - Orange - Yellow - White) 40: Arctic (Black - Dark Blue - Cyan - White)	
`MaskImage` Bitmap, Optional	Optional mask to constrain mapping.	`Nothing`
`MaskRegionBoundary` Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
`IsMaskAllRegionSelected` Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Bitmap` — Colorized image, or Nothing if mapping fails.

EXAMPLE CODE

```
let colored = ImgProc.GrayToColor(grayBitmap,
ImgProcessing.ColorMapType.Jet);
```

27. ImgProc.HexToRGB(HexColorStr, R, G, B) Returns: Void

DESCRIPTION

Converts hexadecimal color string to RGB byte values.

SYNTAX

```
ImgProc.HexToRGB(HexColorStr, R, G, B);
```

PARAMETERS

Parameter	Description	Default
HexColorStr String	Hex color string (e.g., "#FF0000").	–
R Byte	Output red component (0-255).	–
G Byte	Output green component (0-255).	–
B Byte	Output blue component (0-255).	–

RETURN VALUE

`Void` — Void; returns RGB values through ByRef parameters.

EXAMPLE CODE

```
Dim r, g, b As Byte: ImgProc.HexToRGB("#FF0000", r, g, b);
```

28. ImgProc.Hide_Outlines(SourceImage, TargetColor_R, TargetColor_G, TargetColor_B, IsMostFrequentColor, RepeatCount, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Removes or hides outlines of specified color from image.

SYNTAX

```
let result = ImgProc.Hide_Outlines(SourceImage, TargetColor_R, TargetColor_G, TargetColor_B, IsMostFrequentColor, RepeatCount, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
`SourceImage` Bitmap	Source image to modify. Type: Bitmap; required; passed by reference.	–
`TargetColor_R` Byte	Red channel of outline color.	–
`TargetColor_G` Byte	Green channel of outline color.	–
`TargetColor_B` Byte	Blue channel of outline color.	–
`IsMostFrequentColor` Boolean, Optional	Whether to use most frequent color for replacement.	`True`
`RepeatCount` Integer, Optional	Number of iterations to repeat the outline hiding process (1–50).	`1`
`MaskImage` Bitmap, Optional	Optional mask to constrain operation.	`Nothing`
`MaskRegionBoundary` Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
`IsMaskAllRegionSelected` Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Bitmap` — Modified image with hidden outlines, or Nothing if operation fails.

EXAMPLE CODE

```
let hidden = ImgProc.Hide_Outlines(bitmap, 0, 0, 0, true, 3);
```

29. ImgProc.HSITORGB(Hue, Saturation255, Intensity255, Red,

Green, Blue)

Returns: Void

DESCRIPTION

Converts HSI to RGB color space.

SYNTAX

```
ImgProc.HSITORGB(Hue, Saturation255, Intensity255, Red, Green, Blue);
```

PARAMETERS

Parameter	Description	Default
Hue Integer	Hue value.	–
Saturation255 Single	Saturation (0-255).	–
Intensity255 Single	Intensity (0-255).	–
Red Integer	Output red component. Type: Integer; required; passed by reference.	–
Green Integer	Output green component. Type: Integer; required; passed by reference.	–
Blue Integer	Output blue component. Type: Integer; required; passed by reference.	–

RETURN VALUE

Void

EXAMPLE CODE

```
ImgProc.HSITORGB(180, 128, 200, red, green, blue);
```

30. ImgProc.HSVtoRGB(h, s, v, r, g, b) Returns: Void

DESCRIPTION

Converts HSV to RGB color space

SYNTAX

```
ImgProc.HSVtoRGB(h, s, v, r, g, b);
```

PARAMETERS

Parameter	Description	Default
h Single	Hue value.	–
s Single	Saturation value.	–
v Single	Value component.	–
r Byte	Output red component. Type: Byte; required; passed by reference.	–
g Byte	Output green component. Type: Byte; required; passed by reference.	–
b Byte	Output blue component. Type: Byte; required; passed by reference.	–

RETURN VALUE

Void

EXAMPLE CODE

```
ImgProc.HSVtoRGB(180, 0.5, 0.75, r, g, b);
```

31. ImgProc.HueSaturationIntensity(original, hueShift, satShift, intShift, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Adjusts hue, saturation, and intensity values of the image.

SYNTAX

```
let result = ImgProc.HueSaturationIntensity(original, hueShift, satShift, intShift, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
original Bitmap	Source image to adjust.	–
hueShift Integer	Hue shift (-180 to 180).	–
satShift Integer	Saturation shift (-100 to 100).	–
intShift Integer	Intensity shift (-100 to 100).	–
MaskImage Image, Optional	Optional mask to constrain adjustment.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Bitmap` — Hue-saturation-intensity adjusted image, or Nothing if processing fails.

EXAMPLE CODE

```
let hsi = ImgProc.HueSaturationIntensity(bitmap, 30, 20, -10);
```

32. ImgProc.IsBinaryImage(SourceImage) Returns: Boolean

DESCRIPTION

Checks if image is binary (contains only black and white pixels).

SYNTAX

```
let result = ImgProc.IsBinaryImage(SourceImage);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Image to check.	–

RETURN VALUE

Boolean — True if image is binary, False otherwise.

EXAMPLE CODE

```
let isBinary = ImgProc.IsBinaryImage(bitmap);
```

33. ImgProc.IsBWPixel(red, green, blue) Returns: Boolean

DESCRIPTION

Checks if RGB pixel values represent black or white.

SYNTAX

```
let result = ImgProc.IsBWPixel(red, green, blue);
```

PARAMETERS

Parameter	Description	Default
red Byte	Red component (0-255).	–
green Byte	Green component (0-255).	–
blue Byte	Blue component (0-255).	–

RETURN VALUE

Boolean — True if pixel is black or white, False otherwise.

EXAMPLE CODE

```
let isBW = ImgProc.IsBWPixel(0, 0, 0);
```

34. ImgProc.IsGrayImage(SourceImage) Returns: Boolean

DESCRIPTION

Checks if image is grayscale (all color channels equal).

SYNTAX

```
let result = ImgProc.IsGrayImage(SourceImage);
```

PARAMETERS

Parameter	Description	Default
`SourceImage` Bitmap	Image to check.	–

RETURN VALUE

Boolean — True if image is grayscale, False otherwise.

EXAMPLE CODE

```
let isGray = ImgProc.IsGrayImage(bitmap);
```

35. ImgProc.Maximum(rR, rG, rB) Returns: Single

DESCRIPTION

Returns maximum value among three color components.

SYNTAX

```
let result = ImgProc.Maximum(rR, rG, rB);
```

PARAMETERS

Parameter	Description	Default
`rR` Single	Red component value.	–
`rG` Single	Green component value.	–
`rB` Single	Blue component value.	–

RETURN VALUE

Single — Maximum value among the three components.

EXAMPLE CODE

```
let max = ImgProc.Maximum(255.0, 128.0, 200.0);
```

36. ImgProc.MergeChannels(SourceImage, ColorSpace, Image1, Image2, Image3, maskImage, maskRegionBoundary, isMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Merges three channel images using specified color space.

SYNTAX

```
let result = ImgProc.MergeChannels(SourceImage, ColorSpace, Image1, Image2, Image3, maskImage, maskRegionBoundary, isMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Image	Original source image for background preservation.	–
ColorSpace String	Target color space ('RGB', 'HSI', or 'LAB').	–
Image1 Bitmap	First channel image.	–
Image2 Bitmap	Second channel image.	–
Image3 Bitmap	Third channel image.	–
maskImage Bitmap, Optional	Optional mask to constrain operation.	`Nothing`
maskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
isMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Bitmap` — Merged image in specified color space, or Nothing if merging fails.

EXAMPLE CODE

```
let merged = ImgProc.MergeChannels(bitmap, "RGB", ch1, ch2, ch3);
```

37. ImgProc.Minimum(rR, rG, rB) Returns: Single

DESCRIPTION

Returns minimum value among three color components.

SYNTAX

```
let result = ImgProc.Minimum(rR, rG, rB);
```

PARAMETERS

Parameter	Description	Default
rR Single	Red component value.	–
rG Single	Green component value.	–
rB Single	Blue component value.	–

RETURN VALUE

`Single` — Minimum value among the three components.

EXAMPLE CODE

```
let min = ImgProc.Minimum(255.0, 128.0, 200.0);
```

38. ImgProc.MultiStepThresholding(SourceImage, LevelCount_Red, LevelCount_Green, LevelCount_Blue, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Applies multi-step thresholding to separate color channels into quantized levels.

SYNTAX

```
let result = ImgProc.MultiStepThresholding(SourceImage, LevelCount_Red, LevelCount_Green, LevelCount_Blue, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to threshold.	–
LevelCount_Red	Number of quantization levels	

LevelCount_Red Integer	Number of quantization levels for red channel.	–
LevelCount_Green Integer	Number of quantization levels for green channel.	–
LevelCount_Blue Integer	Number of quantization levels for blue channel.	–
MaskImage Bitmap, Optional	Optional mask to constrain thresholding.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Thresholded image, or Nothing if processing fails.

EXAMPLE CODE

```
let thresholded = ImgProc.MultiStepThresholding(bitmap, 3, 3, 3);
```

39. ImgProc.RemoveTransparency(SourceImage) Returns: Bitmap

DESCRIPTION

Removes alpha channel transparency from image.

SYNTAX

```
let result = ImgProc.RemoveTransparency(SourceImage);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to process.	–

RETURN VALUE

Bitmap — Image without transparency, or Nothing if removal fails.

EXAMPLE CODE

```
let noAlpha = ImgProc.RemoveTransparency(bitmap);
```

40. ImgProc.RGBToARGBInteger(red, green, blue)

Returns: Integer

DESCRIPTION

Converts RGB byte values to ARGB integer format with full alpha.

SYNTAX

```
let result = ImgProc.RGBToARGBInteger(red, green, blue);
```

PARAMETERS

Parameter	Description	Default
red Integer	Red component (0-255).	–
green Integer	Green component (0-255).	–
blue Integer	Blue component (0-255).	–

RETURN VALUE

Integer — ARGB integer value, or 0 if conversion fails.

EXAMPLE CODE

```
let argb = ImgProc.RGBToARGBInteger(255, 128, 64);
```

41. ImgProc.RGBToCMYK(r, g, b, IsCompute_Cyan, IsCompute_Magenta, IsCompute_Yellow)

Returns: Void

DESCRIPTION

Converts RGB to CMYK color space.

SYNTAX

```
ImgProc.RGBToCMYK(r, g, b, IsCompute_Cyan, IsCompute_Magenta, IsCompute_Yellow);
```

PARAMETERS

Parameter	Description	Default
r	Red component value.	–

Byte		
g Byte	Green component value.	–
b Byte	Blue component value.	–
IsCompute_Cyan Boolean	Compute cyan component.	–
IsCompute_Magenta Boolean	Compute magenta component.	–
IsCompute_Yellow Boolean	Compute yellow component.	–

RETURN VALUE

Void

EXAMPLE CODE

```
ImgProc.RGBToCMYK(255, 128, 64, true, true, true);
```

42. ImgProc.RGBToColor(RedInByte, GreenInByte, BlueInByte)

Returns: Color

DESCRIPTION

Converts RGB byte values to Color object.

SYNTAX

```
let result = ImgProc.RGBToColor(RedInByte, GreenInByte, BlueInByte);
```

PARAMETERS

Parameter	Description	Default
RedInByte Integer	Red component (0-255).	–
GreenInByte Integer	Green component (0-255).	–
BlueInByte Integer	Blue component (0-255).	–

RETURN VALUE

`Color` — Color object, or Color.Black if conversion fails.

EXAMPLE CODE

```
let color = ImgProc.RGBToColor(255, 100, 50);
```

43. ImgProc.RGBToColorHex(RedValue, GreenValue, BlueValue)

Returns: String

DESCRIPTION

Converts RGB values to hexadecimal color string.

SYNTAX

```
let result = ImgProc.RGBToColorHex(RedValue, GreenValue, BlueValue);
```

PARAMETERS

Parameter	Description	Default
RedValue Integer	Red component value.	–
GreenValue Integer	Green component value.	–
BlueValue Integer	Blue component value.	–

RETURN VALUE

`String` — Hexadecimal color string, or empty string if conversion fails.

EXAMPLE CODE

```
let hexColor = ImgProc.RGBToColorHex(255, 0, 0);
```

44. ImgProc.RGBToGChromaticity(_RedValue, _GreenValue, _BlueValue)

Returns: Void

DESCRIPTION

Converts RGB to green chromaticity coordinate.

SYNTAX

```
ImgProc.RGBToGChromaticity(_RedValue, _GreenValue, _BlueValue);
```

PARAMETERS

Parameter	Description	Default
_RedValue Integer	Red channel value.	–
_GreenValue Integer	Green channel value.	–
_BlueValue Integer	Blue channel value.	–

RETURN VALUE

Void — None.

EXAMPLE CODE

```
ImgProc.RGBToGChromaticity(255, 128, 0);
```

45. ImgProc.RGBToGChromaticity_Shared(RedValue, GreenValue, BlueValue, RGChromaticity_Red, RGChromaticity_Green, RGChromaticity_Blue)

Returns: Void

DESCRIPTION

Converts RGB to rg chromaticity values (shared version).

SYNTAX

```
ImgProc.RGBToGChromaticity_Shared(RedValue, GreenValue, BlueValue, RGChromaticity_Red, RGChromaticity_Green, RGChromaticity_Blue);
```

PARAMETERS

Parameter	Description	Default
RedValue Integer	Red channel value.	–
GreenValue Integer	Green channel value.	–

BlueValue Integer	Blue channel value.	–
RGChromaticity_Red Byte	Output rg chromaticity red value. Type: Byte; required; passed by reference.	–
RGChromaticity_Green Byte	Output rg chromaticity green value. Type: Byte; required; passed by reference.	–
RGChromaticity_Blue Byte	Output rg chromaticity blue value. Type: Byte; required; passed by reference.	–

RETURN VALUE

`Void` — None.

EXAMPLE CODE

```
ImgProc.RGBToGChromaticity_Shared(255, 128, 0, outR, outG, outB);
```

46. ImgProc.RGBToGray(r, g, b) Returns: Byte

DESCRIPTION

Converts RGB to grayscale byte value using standard formula.

SYNTAX

```
let result = ImgProc.RGBToGray(r, g, b);
```

PARAMETERS

Parameter	Description	Default
r Integer	Red component (0-255).	–
g Integer	Green component (0-255).	–
b Integer	Blue component (0-255).	–

RETURN VALUE

`Byte` — Grayscale byte value (0-255), or 0 if conversion fails.

EXAMPLE CODE

```
let grayByte = ImgProc.RGBToGray(255, 100, 50);
```

47. ImgProc.RGBToGray_ByRealCalc(r, g, b) Returns: Integer

DESCRIPTION

Converts RGB to grayscale using precise luminosity calculation.

SYNTAX

```
let result = ImgProc.RGBToGray_ByRealCalc(r, g, b);
```

PARAMETERS

Parameter	Description	Default
`r` Integer	Red component (0-255).	–
`g` Integer	Green component (0-255).	–
`b` Integer	Blue component (0-255).	–

RETURN VALUE

Integer — Grayscale value (0-255), or 0 if conversion fails.

EXAMPLE CODE

```
let grayValue = ImgProc.RGBToGray_ByRealCalc(255, 100, 50);
```

48. ImgProc.RGBToHSI(r, g, b) Returns: Void

DESCRIPTION

Converts RGB to HSI color space.

SYNTAX

```
ImgProc.RGBToHSI(r, g, b);
```

PARAMETERS

Parameter	Description	Default

r Byte	Red component value.	–
g Byte	Green component value.	–
b Byte	Blue component value.	–

RETURN VALUE

Void

EXAMPLE CODE

```
ImgProc.RGBToHSI(255, 128, 64);
```

49. ImgProc.RGBToHSI_Parallel(r, g, b, Hue, Saturation, Intensity)

Returns: Void

DESCRIPTION

Converts RGB to HSI using parallel method.

SYNTAX

```
ImgProc.RGBToHSI_Parallel(r, g, b, Hue, Saturation, Intensity);
```

PARAMETERS

Parameter	Description	Default
r Integer	Red component value.	–
g Integer	Green component value.	–
b Integer	Blue component value.	–
Hue Single	Output hue value. Type: Single; required; passed by reference.	–
Saturation Single	Output saturation value. Type: Single; required; passed by reference.	–

Intensity Single	Output intensity value. Type: Single; required; passed by reference.	–

RETURN VALUE

Void

EXAMPLE CODE

```
ImgProc.RGBToHSI_Parallel(255, 128, 64, hue, sat, int);
```

50. ImgProc.RGBToLab(r, g, b) Returns: Void

DESCRIPTION

Converts RGB to CIELab color space.

SYNTAX

```
ImgProc.RGBToLab(r, g, b);
```

PARAMETERS

Parameter	Description	Default
r Byte	Red component value.	–
g Byte	Green component value.	–
b Byte	Blue component value.	–

RETURN VALUE

Void

EXAMPLE CODE

```
ImgProc.RGBToLab(255, 128, 64);
```

51. ImgProc.RGBToLab_OriginalValue(r, g, b, LabL, Laba, Labb)

Returns: Void

DESCRIPTION

Converts RGB to CIELab with original value output.

SYNTAX

```
ImgProc.RGBToLab_OriginalValue(r, g, b, LabL, Laba, Labb);
```

PARAMETERS

Parameter	Description	Default
r Byte	Red component value.	–
g Byte	Green component value.	–
b Byte	Blue component value.	–
LabL Double	Output L component. Type: Double; required; passed by reference.	–
Laba Double	Output a component. Type: Double; required; passed by reference.	–
Labb Double	Output b component. Type: Double; required; passed by reference.	–

RETURN VALUE

Void

EXAMPLE CODE

```
ImgProc.RGBToLab_OriginalValue(255, 128, 64, labL, labA, labB);
```

52. ImgProc.RGBToLab_Parallel(r, g, b, Out_L, Out_a, Out_b)

Returns: Void

DESCRIPTION

Converts RGB to CIELab using parallel method.

SYNTAX

```
ImgProc.RGBToLab_Parallel(r, g, b, Out_L, Out_a, Out_b);
```

PARAMETERS

PARAMETERS

Parameter	Description	Default
`r` Byte	Red component value.	–
`g` Byte	Green component value.	–
`b` Byte	Blue component value.	–
`Out_L` Byte	Output L component. Type: Byte; required; passed by reference.	–
`Out_a` Byte	Output a component. Type: Byte; required; passed by reference.	–
`Out_b` Byte	Output b component. Type: Byte; required; passed by reference.	–

RETURN VALUE

Void

EXAMPLE CODE

```
ImgProc.RGBToLab_Parallel(255, 128, 64, outL, outA, outB);
```

53. ImgProc.Split_a(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Splits a (color opponent) channel in Lab color space.

SYNTAX

```
let result = ImgProc.Split_a(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
`SourceImage` Bitmap	Source image to decompose. Type: Bitmap; required; passed by reference.	–

MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — a channel image, or Nothing if decomposition fails.

EXAMPLE CODE

```
let aChannel = ImgProc.Split_a(bitmap);
```

54. ImgProc.Split_b(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Splits b (color opponent) channel in Lab color space.

SYNTAX

```
let result = ImgProc.Split_b(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to decompose. Type: Bitmap; required; passed by reference.	–
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

`Bitmap` — b channel image, or Nothing if decomposition fails.

EXAMPLE CODE

```
let bChannel = ImgProc.Split_b(bitmap);
```

55. ImgProc.Split_Blue(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Extracts blue channel from color image as grayscale.

SYNTAX

```
let result = ImgProc.Split_Blue(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source color image. Type: Bitmap; required; passed by reference.	–
MaskImage Bitmap, Optional	Optional mask to constrain extraction.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Bitmap` — Blue channel as grayscale image, or Nothing if extraction fails.

EXAMPLE CODE

```
let blueChannel = ImgProc.Split_Blue(bitmap);
```

56. ImgProc.Split_Cyan(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Splits C (cyan) channel in CMYK decomposition.

SYNTAX

```
let result = ImgProc.Split_Cyan(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to decompose. Type: Bitmap; required; passed by reference.	–
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — C channel image, or Nothing if decomposition fails.

EXAMPLE CODE

```
let cChannel = ImgProc.Split_Cyan(bitmap);
```

57. ImgProc.Split_Green(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Extracts green channel from color image as grayscale.

SYNTAX

```
let result = ImgProc.Split_Green(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default

SourceImage Bitmap	Source color image. Type: Bitmap; required; passed by reference.	–
MaskImage Bitmap, Optional	Optional mask to constrain extraction.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Green channel as grayscale image, or Nothing if extraction fails.

EXAMPLE CODE

```
let greenChannel = ImgProc.Split_Green(bitmap);
```

58. ImgProc.Split_Hue(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Extracts hue channel from HSI color space.

SYNTAX

```
let result = ImgProc.Split_Hue(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source color image. Type: Bitmap; required; passed by reference.	–
MaskImage Bitmap, Optional	Optional mask to constrain extraction.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected	Whether entire region is	True

	Boolean, Optional	selected.

RETURN VALUE

`Bitmap` — Hue channel as image, or Nothing if extraction fails.

EXAMPLE CODE

```
let hueChannel = ImgProc.Split_Hue(bitmap);
```

59. ImgProc.Split_Intensity(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Extracts intensity channel from HSI color space.

SYNTAX

```
let result = ImgProc.Split_Intensity(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source color image. Type: Bitmap; required; passed by reference.	–
MaskImage Bitmap, Optional	Optional mask to constrain extraction.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Bitmap` — Intensity channel as image, or Nothing if extraction fails.

EXAMPLE CODE

```
let intensity = ImgProc.Split_Intensity(bitmap);
```

60. ImgProc.Split_KBlack(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Splits K (black) channel in CMYK decomposition.

SYNTAX

```
let result = ImgProc.Split_KBlack(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to decompose. Type: Bitmap; required; passed by reference.	–
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — K channel image, or Nothing if decomposition fails.

EXAMPLE CODE

```
let kChannel = ImgProc.Split_KBlack(bitmap);
```

61. ImgProc.Split_L(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Splits L (lightness) channel in Lab color space.

SYNTAX

```
let result = ImgProc.Split_L(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to decompose. Type: Bitmap; required; passed by reference.	–
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — L channel image, or Nothing if decomposition fails.

EXAMPLE CODE

```
let lChannel = ImgProc.Split_L(bitmap);
```

62. ImgProc.Split_Magenta(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Splits M (magenta) channel in CMYK decomposition.

SYNTAX

```
let result = ImgProc.Split_Magenta(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to decompose. Type: Bitmap; required; passed by reference.	–
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing

MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — M channel image, or Nothing if decomposition fails.

EXAMPLE CODE

```
let mChannel = ImgProc.Split_Magenta(bitmap);
```

63. ImgProc.Split_Red(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Extracts red channel from color image as grayscale.

SYNTAX

```
let result = ImgProc.Split_Red(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source color image. Type: Bitmap; required; passed by reference.	–
MaskImage Bitmap, Optional	Optional mask to constrain extraction.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Red channel as grayscale image, or Nothing if extraction fails.

EXAMPLE CODE

```
let redChannel = ImgProc.Split_Red(bitmap);
```

64. ImgProc.Split_rgChromaticityBlue(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Splits rg chromaticity blue channel.

SYNTAX

```
let result = ImgProc.Split_rgChromaticityBlue(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to decompose. Type: Bitmap; required; passed by reference.	–
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — rg chromaticity blue channel image, or Nothing if splitting fails.

EXAMPLE CODE

```
let rgBlue = ImgProc.Split_rgChromaticityBlue(bitmap);
```

65. ImgProc.Split_rgChromaticityGreen(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Splits rg chromaticity green channel.

SYNTAX

```
let result = ImgProc.Split_rgChromaticityGreen(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to decompose. Type: Bitmap; required; passed by reference.	–
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — rg chromaticity green channel image, or Nothing if splitting fails.

EXAMPLE CODE

```
let rgGreen = ImgProc.Split_rgChromaticityGreen(bitmap);
```

66. ImgProc.Split_rgChromaticityRed(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Splits rg chromaticity red channel.

SYNTAX

```
let result = ImgProc.Split_rgChromaticityRed(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to decompose. Type: Bitmap; required; passed by reference.	–

MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — rg chromaticity red channel image, or Nothing if splitting fails.

EXAMPLE CODE

```
let rgRed = ImgProc.Split_rgChromaticityRed(bitmap);
```

67. ImgProc.Split_Saturation(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Extracts saturation channel from HSI color space.

SYNTAX

```
let result = ImgProc.Split_Saturation(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source color image. Type: Bitmap; required; passed by reference.	–
MaskImage Bitmap, Optional	Optional mask to constrain extraction.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Saturation channel as image, or Nothing if extraction fails.

EXAMPLE CODE

```
let saturation = ImgProc.Split_Saturation(bitmap);
```

68. ImgProc.Split_Yellow(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Splits Y (yellow) channel in CMYK decomposition.

SYNTAX

```
let result = ImgProc.Split_Yellow(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to decompose. Type: Bitmap; required; passed by reference.	–
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Y channel image, or Nothing if decomposition fails.

EXAMPLE CODE

```
let yChannel = ImgProc.Split_Yellow(bitmap);
```

69. ImgProc.SplitToning(SourceImage, ShadowHue, HighlightHue, ToneStrength, TonalBalance, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Applies dual-tone color grading to shadows and highlights independently using a high-speed 256-entry luminance LUT.

SYNTAX

```
let result = ImgProc.SplitToning(SourceImage, ShadowHue, HighlightHue, ToneStrength, TonalBalance, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to process.	–
ShadowHue Integer, Optional	Color hue angle in degrees for shadows (0-360).	210
HighlightHue Integer, Optional	Color hue angle in degrees for highlights (0-360).	35
ToneStrength Single, Optional	Overall color toning intensity (0.0-1.0).	0.4F
TonalBalance Integer, Optional	Balance shift between shadow and highlight tints (-100 to 100).	0
IsShowProgress Boolean, Optional	Whether to show processing progress.	True
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

`Bitmap` — Processed Bitmap with Split Toning applied, or Nothing if processing fails.

EXAMPLE CODE

```
let tonedBmp = ImgProc.SplitToning(bitmap, 210, 35, 0.4F, 0);
```

Data Extraction & Analysis Actions 75 functions

1. ImgProc.AreImagesIdentical(img1, img2) Returns: Boolean

DESCRIPTION

Checks if two images are identical.

SYNTAX

```
let result = ImgProc.AreImagesIdentical(img1, img2);
```

PARAMETERS

Parameter	Description	Default
img1 Bitmap	First image to compare.	–
img2 Bitmap	Second image to compare.	–

RETURN VALUE

Boolean — True if images are identical, False otherwise.

EXAMPLE CODE

```
let identical = ImgProc.AreImagesIdentical(bitmap1, bitmap2);
```

2. ImgProc.Calculate_AverageColorValue(SourceImage, TotalPixelsCount, AnalysisInterval, ColorChannel, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Single

DESCRIPTION

Calculates average color value of specified channel.

SYNTAX

```
let result = ImgProc.Calculate_AverageColorValue(SourceImage, TotalPixelsCount, AnalysisInterval, ColorChannel, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
`SourceImage` Bitmap	Source image to analyze.	–
`TotalPixelsCount` Long	Output total pixels analyzed. Type: Long; required; passed by reference.	–
`AnalysisInterval` Integer	Sampling interval for analysis.	–
`ColorChannel` String	Color channel to analyze. Supported values (case-insensitive): "gray": Gray level "red": Red "green": Green "blue": Blue "hue": Hue "saturation": Saturation "intensity": Intensity "cyan": Cyan "magenta": Magenta "yellow": Yellow "k black": K (Black) of CMYK "lab l": CIE Lab L (Lightness) "lab a": CIE Lab a (Green-Red) "lab b": CIE Lab b (Blue-Yellow) "rg chromaticity red": rg chromaticity Red "rg chromaticity green": rg chromaticity Green "rg chromaticity blue": rg chromaticity Blue Any other value: the function returns -1	–
`MaskImage` Bitmap, Optional	Optional mask to constrain analysis.	`Nothing`
`MaskRegionBoundary` Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
`IsMaskAllRegionSelected` Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Single` — Average color value of the specified channel, or 0 if calculation fails.

EXAMPLE CODE

```
let avgRed = ImgProc.Calculate_AverageColorValue(bitmap, totPix, 1,
"R");
```

3. ImgProc.Calculate_AverageColorValue_a(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Single

DESCRIPTION

Calculates average a (green-red) value in CIELab.

SYNTAX

```
let result = ImgProc.Calculate_AverageColorValue_a(SourceImage, T
otalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary,
IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to analyze.	–
TotalPixelsCount Long	Output total pixels analyzed. Type: Long; required; passed by reference.	–
AnalysisInterval Integer	Sampling interval for analysis.	–
MaskImage Bitmap, Optional	Optional mask to constrain analysis.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Single` — Average a (green-red) value, or 0 if calculation fails.

EXAMPLE CODE

```
let avgA = ImgProc.Calculate_AverageColorValue_a(bitmap, totPix,
1);
```

4. ImgProc.Calculate_AverageColorValue_b(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Single

DESCRIPTION

Calculates average b (blue-yellow) value in CIELab.

SYNTAX

```
let result = ImgProc.Calculate_AverageColorValue_b(SourceImage, T
otalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary,
IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to analyze.	–
TotalPixelsCount Long	Output total pixels analyzed. Type: Long; required; passed by reference.	–
AnalysisInterval Integer	Sampling interval for analysis.	–
MaskImage Bitmap, Optional	Optional mask to constrain analysis.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Single — Average b (blue-yellow) value, or 0 if calculation fails.

EXAMPLE CODE

```
let avgB = ImgProc.Calculate_AverageColorValue_b(bitmap, totPix,
1);
```

5. ImgProc.Calculate_AverageColorValue_Blue(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Single

DESCRIPTION

Calculates average blue channel value.

SYNTAX

```
let result = ImgProc.Calculate_AverageColorValue_Blue(SourceImag
e, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBound
ary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to analyze.	–
TotalPixelsCount Long	Output total pixels analyzed. Type: Long; required; passed by reference.	–
AnalysisInterval Integer	Sampling interval for analysis.	–
MaskImage Bitmap, Optional	Optional mask to constrain analysis.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Single — Average blue channel value, or 0 if calculation fails.

EXAMPLE CODE

```
let avgBlue = ImgProc.Calculate_AverageColorValue_Blue(bitmap,
totPix, 1);
```

6. ImgProc.Calculate_AverageColorValue_Cyan(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Single

DESCRIPTION

Calculates average cyan channel value.

SYNTAX

```
let result = ImgProc.Calculate_AverageColorValue_Cyan(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to analyze.	–
TotalPixelsCount Long	Output total pixels analyzed. Type: Long; required; passed by reference.	–
AnalysisInterval Integer	Sampling interval for analysis.	–
MaskImage Bitmap, Optional	Optional mask to constrain analysis.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Single` — Average cyan value, or 0 if calculation fails.

EXAMPLE CODE

```
let avgCyan = ImgProc.Calculate_AverageColorValue_Cyan(bitmap, totPix, 1);
```

7. ImgProc.Calculate_AverageColorValue_Gray(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Single

DESCRIPTION

Calculates average grayscale value.

SYNTAX

```
let result = ImgProc.Calculate_AverageColorValue_Gray(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to analyze.	–
TotalPixelsCount Long	Output total pixels analyzed. Type: Long; required; passed by reference.	–
AnalysisInterval Integer	Sampling interval for analysis.	–
MaskImage Bitmap, Optional	Optional mask to constrain analysis.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Single — Average grayscale value, or 0 if calculation fails.

EXAMPLE CODE

```
let avgGray = ImgProc.Calculate_AverageColorValue_Gray(bitmap, totPix, 1);
```

8. ImgProc.Calculate_AverageColorValue_Green(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage,

MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Single

DESCRIPTION

Calculates average green channel value.

SYNTAX

```
let result = ImgProc.Calculate_AverageColorValue_Green(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to analyze.	–
TotalPixelsCount Long	Output total pixels analyzed. Type: Long; required; passed by reference.	–
AnalysisInterval Integer	Sampling interval for analysis.	–
MaskImage Bitmap, Optional	Optional mask to constrain analysis.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Single — Average green channel value, or 0 if calculation fails.

EXAMPLE CODE

```
let avgGreen = ImgProc.Calculate_AverageColorValue_Green(bitmap, totPix, 1);
```

9. ImgProc.Calculate_AverageColorValue_Hue(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Single

Returns: Single

DESCRIPTION

Calculates average hue value.

SYNTAX

```
let result = ImgProc.Calculate_AverageColorValue_Hue(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to analyze.	–
TotalPixelsCount Long	Output total pixels analyzed. Type: Long; required; passed by reference.	–
AnalysisInterval Integer	Sampling interval for analysis.	–
MaskImage Bitmap, Optional	Optional mask to constrain analysis.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Single — Average hue value, or 0 if calculation fails.

EXAMPLE CODE

```
let avgHue = ImgProc.Calculate_AverageColorValue_Hue(bitmap, totPix, 1);
```

10. ImgProc.Calculate_AverageColorValue_Intensity(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Single

DESCRIPTION

Calculates average intensity value.

SYNTAX

```
let result = ImgProc.Calculate_AverageColorValue_Intensity(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to analyze.	–
TotalPixelsCount Long	Output total pixels analyzed. Type: Long; required; passed by reference.	–
AnalysisInterval Integer	Sampling interval for analysis.	–
MaskImage Bitmap, Optional	Optional mask to constrain analysis.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Single` — Average intensity value, or 0 if calculation fails.

EXAMPLE CODE

```
let avgInt = ImgProc.Calculate_AverageColorValue_Intensity(bitmap, totPix, 1);
```

11. ImgProc.Calculate_AverageColorValue_KBlack(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Single

DESCRIPTION

Calculates average black (K) channel value in CMYK.

SYNTAX

```
let result = ImgProc.Calculate_AverageColorValue_KBlack(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to analyze.	–
TotalPixelsCount Long	Output total pixels analyzed. Type: Long; required; passed by reference.	–
AnalysisInterval Integer	Sampling interval for analysis.	–
MaskImage Bitmap, Optional	Optional mask to constrain analysis.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Single — Average black channel value, or 0 if calculation fails.

EXAMPLE CODE

```
let avgK = ImgProc.Calculate_AverageColorValue_KBlack(bitmap, totPix, 1);
```

12. ImgProc.Calculate_AverageColorValue_L(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Single

DESCRIPTION

Calculates average L (lightness) value in CIELab.

SYNTAX

```
let result = ImgProc.Calculate_AverageColorValue_L(SourceImage, T
```

```
otalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary,
IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to analyze.	–
TotalPixelsCount Long	Output total pixels analyzed. Type: Long; required; passed by reference.	–
AnalysisInterval Integer	Sampling interval for analysis.	–
MaskImage Bitmap, Optional	Optional mask to constrain analysis.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Single — Average L (lightness) value, or 0 if calculation fails.

EXAMPLE CODE

```
let avgL = ImgProc.Calculate_AverageColorValue_L(bitmap, totPix,
1);
```

13. ImgProc.Calculate_AverageColorValue_Magenta(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Single

DESCRIPTION

Calculates average magenta channel value.

SYNTAX

```
let result = ImgProc.Calculate_AverageColorValue_Magenta(SourceIm
age, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBou
ndary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
`SourceImage` Bitmap	Source image to analyze.	–
`TotalPixelsCount` Long	Output total pixels analyzed. Type: Long; required; passed by reference.	–
`AnalysisInterval` Integer	Sampling interval for analysis.	–
`MaskImage` Bitmap, Optional	Optional mask to constrain analysis.	`Nothing`
`MaskRegionBoundary` Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
`IsMaskAllRegionSelected` Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Single` — Average magenta value, or 0 if calculation fails.

EXAMPLE CODE

```
let avgMag = ImgProc.Calculate_AverageColorValue_Magenta(bitmap, totPix, 1);
```

14. ImgProc.Calculate_AverageColorValue_MultipleChannels(SourceImage, ShapeMode, X1, Y1, X2, Y2, ReturnRed, ReturnGreen, ReturnBlue, ReturnGray, ReturnHue, ReturnSat, ReturnIntensity, ReturnCyan, ReturnMagenta, ReturnYellow, ReturnK, ReturnLab_L, ReturnLab_a, ReturnLab_b, ReturnRG_Red, ReturnRG_Green, ReturnRG_Blue)

Returns: Void

DESCRIPTION

Calculates multiple color channel averages simultaneously.

SYNTAX

```
ImgProc.Calculate_AverageColorValue_MultipleChannels(SourceImage, ShapeMode, X1, Y1, X2, Y2, ReturnRed, ReturnGreen, ReturnBlue, Re
```

```
turnGray, ReturnHue, ReturnSat, ReturnIntensity, ReturnCyan, Retu
rnMagenta, ReturnYellow, ReturnK, ReturnLab_L, ReturnLab_a, Retur
nLab_b, ReturnRG_Red, ReturnRG_Green, ReturnRG_Blue);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to analyze.	–
ShapeMode Integer	Shape mode for region selection. Supported values: 1: Rectangle region 2: Circle region (inscribed in the X1-X2 range)	–
X1 Integer	First X-coordinate.	–
Y1 Integer	First Y-coordinate.	–
X2 Integer	Second X-coordinate.	–
Y2 Integer	Second Y-coordinate.	–
ReturnRed Byte	Output red channel average. Type: Byte; required; passed by reference.	–
ReturnGreen Byte	Output green channel average. Type: Byte; required; passed by reference.	–
ReturnBlue Byte	Output blue channel average. Type: Byte; required; passed by reference.	–
ReturnGray Byte	Output gray channel average. Type: Byte; required; passed by reference.	–
ReturnHue Single	Output hue average. Type: Single; required; passed by reference.	–
ReturnSat Single	Output saturation average. Type: Single; required; passed by	–

Single	Type: Single; required; passed by reference.	
ReturnIntensity Single	Output intensity average. Type: Single; required; passed by reference.	–
ReturnCyan Single	Output cyan average. Type: Single; required; passed by reference.	–
ReturnMagenta Single	Output magenta average. Type: Single; required; passed by reference.	–
ReturnYellow Single	Output yellow average. Type: Single; required; passed by reference.	–
ReturnK Single		–
ReturnLab_L Single		–
ReturnLab_a Single		–
ReturnLab_b Single		–
ReturnRG_Red Single		–
ReturnRG_Green Single		–
ReturnRG_Blue Single		–

RETURN VALUE

Void

EXAMPLE CODE

```
ImgProc.Calculate_AverageColorValue_MultipleChannels(bitmap, 0, 0, 0, 100, 100, r, g, b, gray, hue, sat, int, cyan, mag, yel);
```

15. ImgProc.Calculate_AverageColorValue_Red(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Single

DESCRIPTION

Calculates average red channel value.

SYNTAX

```
let result = ImgProc.Calculate_AverageColorValue_Red(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to analyze.	–
TotalPixelsCount Long	Output total pixels analyzed. Type: Long; required; passed by reference.	–
AnalysisInterval Integer	Sampling interval for analysis.	–
MaskImage Bitmap, Optional	Optional mask to constrain analysis.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Single — Average red channel value, or 0 if calculation fails.

EXAMPLE CODE

```
let avgRed = ImgProc.Calculate_AverageColorValue_Red(bitmap, totPix, 1);
```

16. ImgProc.Calculate_AverageColorValue_rgChromaticityBlue(SourceImage,

TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Single

DESCRIPTION

Calculates average blue chromaticity value.

SYNTAX

```
let result = ImgProc.Calculate_AverageColorValue_rgChromaticityBlue(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to analyze.	–
TotalPixelsCount Long	Output total pixels analyzed. Type: Long; required; passed by reference.	–
AnalysisInterval Integer	Sampling interval for analysis.	–
MaskImage Bitmap, Optional	Optional mask to constrain analysis.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Single — Average blue chromaticity value, or 0 if calculation fails.

EXAMPLE CODE

```
let avgRgBlue = ImgProc.Calculate_AverageColorValue_rgChromaticityBlue(bitmap, totPix, 1);
```

17. ImgProc.Calculate_AverageColorValue_rgChromaticityGreen(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary,

IsMaskAllRegionSelected)

Returns: Single

DESCRIPTION

Calculates average green chromaticity value.

SYNTAX

```
let result = ImgProc.Calculate_AverageColorValue_rgChromaticityGreen(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to analyze.	–
TotalPixelsCount Long	Output total pixels analyzed. Type: Long; required; passed by reference.	–
AnalysisInterval Integer	Sampling interval for analysis.	–
MaskImage Bitmap, Optional	Optional mask to constrain analysis.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Single — Average green chromaticity value, or 0 if calculation fails.

EXAMPLE CODE

```
let avgRgGreen = ImgProc.Calculate_AverageColorValue_rgChromaticityGreen(bitmap, totPix, 1);
```

18. ImgProc.Calculate_AverageColorValue_rgChromaticityRed(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary,

IsMaskAllRegionSelected)

Returns: Single

DESCRIPTION

Calculates average red chromaticity value.

SYNTAX

```
let result = ImgProc.Calculate_AverageColorValue_rgChromaticityRed(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to analyze.	–
TotalPixelsCount Long	Output total pixels analyzed. Type: Long; required; passed by reference.	–
AnalysisInterval Integer	Sampling interval for analysis.	–
MaskImage Bitmap, Optional	Optional mask to constrain analysis.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Single — Average red chromaticity value, or 0 if calculation fails.

EXAMPLE CODE

```
let avgRgRed = ImgProc.Calculate_AverageColorValue_rgChromaticityRed(bitmap, totPix, 1);
```

19. ImgProc.Calculate_AverageColorValue_Saturation(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Single

DESCRIPTION

Calculates average saturation value.

SYNTAX

```
let result = ImgProc.Calculate_AverageColorValue_Saturation(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to analyze.	–
TotalPixelsCount Long	Output total pixels analyzed. Type: Long; required; passed by reference.	–
AnalysisInterval Integer	Sampling interval for analysis.	–
MaskImage Bitmap, Optional	Optional mask to constrain analysis.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Single` — Average saturation value, or 0 if calculation fails.

EXAMPLE CODE

```
let avgSat = ImgProc.Calculate_AverageColorValue_Saturation(bitmap, totPix, 1);
```

20. ImgProc.Calculate_AverageColorValue_Yellow(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Single

DESCRIPTION

Calculates average yellow channel value.

SYNTAX

```
let result = ImgProc.Calculate_AverageColorValue_Yellow(SourceImage, TotalPixelsCount, AnalysisInterval, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to analyze.	–
TotalPixelsCount Long	Output total pixels analyzed. Type: Long; required; passed by reference.	–
AnalysisInterval Integer	Sampling interval for analysis.	–
MaskImage Bitmap, Optional	Optional mask to constrain analysis.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Single` — Average yellow value, or 0 if calculation fails.

EXAMPLE CODE

```
let avgYellow = ImgProc.Calculate_AverageColorValue_Yellow(bitmap, totPix, 1);
```

21. ImgProc.Calculate_AvgValueByPredefinedColorFormula(SourceImage, TotalPixelsCount, AnalysisInterval, FormulaMethod, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Single

DESCRIPTION

Calculates average using predefined color formula. The result is a 16-bit value combining two channels (low byte | high byte << 8).

SYNTAX

```
let result = ImgProc.Calculate_AvgValueByPredefinedColorFormula(SourceImage, TotalPixelsCount, AnalysisInterval, FormulaMethod, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to analyze.	–
TotalPixelsCount Long	Output total pixels analyzed. Type: Long; required; passed by reference.	–
AnalysisInterval Integer	Sampling interval for analysis.	–
FormulaMethod Integer	Color formula method to use. Supported values: 0: Low byte = Red, High byte = Green 1: Low byte = Red, High byte = Blue 2: Low byte = Green, High byte = Red 3: Low byte = Green, High byte = Blue 4: Low byte = Blue, High byte = Red 5: Low byte = Blue, High byte = Green	–
MaskImage Bitmap, Optional	Optional mask to constrain analysis.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Single` — Average value computed by formula, or 0 if calculation fails.

EXAMPLE CODE

```
let avgFormula =
ImgProc.Calculate_AvgValueByPredefinedColorFormula(bitmap, totPix,
1, 0);
```

22. ImgProc.Calculate_LocalGrayAvgValue(IntegralGrayImageMap(,), LocalBoxSize, CenterX, CenterY)

Returns: Byte

DESCRIPTION

Calculates local grayscale average from integral map.

SYNTAX

```
let result = ImgProc.Calculate_LocalGrayAvgValue(IntegralGrayImag
eMap(,), LocalBoxSize, CenterX, CenterY);
```

PARAMETERS

Parameter	Description	Default
`IntegralGrayImageMap(,)` Long		–
`LocalBoxSize` Integer	Size of local box for averaging.	–
`CenterX` Integer	Center X-coordinate.	–
`CenterY` Integer	Center Y-coordinate.	–

RETURN VALUE

Byte — Local average grayscale value, or 0 if calculation fails.

EXAMPLE CODE

```
let localAvg = ImgProc.Calculate_LocalGrayAvgValue(integralMap, 15,
50, 50);
```

23. ImgProc.Calculate_TotalPixels(SourceImage, SelectShape, X1, Y1, X2, Y2, ColorBW)

Returns: Long

DESCRIPTION

Calculates total pixel count in selected region.

SYNTAX

```
let result = ImgProc.Calculate_TotalPixels(SourceImage, SelectSha
pe, X1, Y1, X2, Y2, ColorBW);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to analyze.	–
SelectShape Integer	Shape mode for region selection. Supported values: 1: Rectangle region Any other value: Circle region	–
X1 Integer	First X-coordinate.	–
Y1 Integer	First Y-coordinate.	–
X2 Integer	Second X-coordinate.	–
Y2 Integer	Second Y-coordinate.	–
ColorBW Integer	Color for counting. Supported values: 0: Count white pixels (255,255,255) Any other value (e.g., 1): Count black pixels (0,0,0)	–

RETURN VALUE

Long — Total pixel count, or 0 if calculation fails.

EXAMPLE CODE

```
let totPix = ImgProc.Calculate_TotalPixels(bitmap, 0, 0, 0, 100,
100, 0);
```

24. ImgProc.Check_ImageSize(SourceImage, MinWidth, MinHeight,

MaxWidth, MaxHeight)

Returns: String

DESCRIPTION

Validates image dimensions against constraints.

SYNTAX

```
let result = ImgProc.Check_ImageSize(SourceImage, MinWidth, MinHeight, MaxWidth, MaxHeight);
```

PARAMETERS

Parameter	Description	Default
`SourceImage` Bitmap	Source image to validate. Type: Bitmap; required; passed by reference.	–
`MinWidth` Integer, Optional	Minimum allowed width.	1
`MinHeight` Integer, Optional	Minimum allowed height.	1
`MaxWidth` Integer, Optional	Maximum allowed width.	10000
`MaxHeight` Integer, Optional	Maximum allowed height.	10000

RETURN VALUE

String — Validation result message, or empty string if validation fails.

EXAMPLE CODE

```
let msg = ImgProc.Check_ImageSize(bitmap, 100, 100, 5000, 5000);
```

25. ImgProc.CompareBitmapImages(b1, b2) Returns: Boolean

DESCRIPTION

Compares two bitmaps for pixel-by-pixel equality.

SYNTAX

```
let result = ImgProc.CompareBitmapImages(b1, b2);
```

PARAMETERS

Parameter	Description	Default
b1 Bitmap	First bitmap to compare.	–
b2 Bitmap	Second bitmap to compare.	–

RETURN VALUE

Boolean — True if bitmaps are identical, False otherwise.

EXAMPLE CODE

```
let isSame = ImgProc.CompareBitmapImages(bmp1, bmp2);
```

26. ImgProc.Compute_DifferentialImage(SourceGrayImage1, SourceGrayImage2, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Computes difference between two images.

SYNTAX

```
let result = ImgProc.Compute_DifferentialImage(SourceGrayImage1, SourceGrayImage2, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceGrayImage1 Bitmap	First grayscale image. Type: Bitmap; required; passed by reference.	–
SourceGrayImage2 Bitmap	Second grayscale image. Type: Bitmap; required; passed by reference.	–
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing

`IsMaskAllRegionSelected` Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Differential image, or Nothing if operation fails.

EXAMPLE CODE

```
let diff = ImgProc.Compute_DifferentialImage(grayBitmap1,
grayBitmap2);
```

27. ImgProc.Convert_GrayByteArray_To_RGBImage(SrcByteArray(,))

Returns: Bitmap

DESCRIPTION

Converts grayscale byte array to RGB image.

SYNTAX

```
let result = ImgProc.Convert_GrayByteArray_To_RGBImage(SrcByteArr
ay(,));
```

PARAMETERS

Parameter	Description	Default
`SrcByteArray(,)` Byte		–

RETURN VALUE

Bitmap — RGB image converted from byte array, or Nothing if conversion fails.

EXAMPLE CODE

```
let image = ImgProc.Convert_GrayByteArray_To_RGBImage(grayArray);
```

28. ImgProc.Convert_GrayByteArray_To_RGBImage_Int(SrcByteArray(,))

Returns: Bitmap

DESCRIPTION

Converts grayscale integer array to RGB image.

SYNTAX

```
let result = ImgProc.Convert_GrayByteArray_To_RGBImage_Int(SrcByteArray(,));
```

PARAMETERS

Parameter	Description	Default
SrcByteArray(,) Integer		–

RETURN VALUE

Bitmap — RGB image converted from integer array, or Nothing if conversion fails.

EXAMPLE CODE

```
let image = ImgProc.Convert_GrayByteArray_To_RGBImage_Int(intArray);
```

29. ImgProc.Convert_GrayImage_To_GrayByteArray(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Byte(,)

DESCRIPTION

Extracts grayscale image pixels into byte array.

SYNTAX

```
let result = ImgProc.Convert_GrayImage_To_GrayByteArray(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to extract.	–
MaskImage Bitmap, Optional	Optional mask to constrain extraction.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Byte(,) — 2D byte array (height x width), or Nothing if extraction fails.

EXAMPLE CODE

```
let grayArray = ImgProc.Convert_GrayImage_To_GrayByteArray(bitmap);
```

30. ImgProc.Convert_RGBByteArray_To_RGBImage(SrcByteArray(,,))

Returns: Bitmap

DESCRIPTION

Converts RGB byte array to RGB image.

SYNTAX

```
let result = ImgProc.Convert_RGBByteArray_To_RGBImage(SrcByteArray(,,));
```

PARAMETERS

Parameter	Description	Default
`SrcByteArray(,,)` Byte		–

RETURN VALUE

Bitmap — RGB image converted from byte array, or Nothing if conversion fails.

EXAMPLE CODE

```
let image = ImgProc.Convert_RGBByteArray_To_RGBImage(rgbArray);
```

31. ImgProc.Convert_RGBImage_To_GrayByteArray(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Byte(,)

DESCRIPTION

Converts RGB image pixels to grayscale byte array.

SYNTAX

```
let result = ImgProc.Convert_RGBImage_To_GrayByteArray(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default

SourceImage Bitmap	Source image to convert.	–
MaskImage Bitmap, Optional	Optional mask to constrain conversion.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Byte(,) — 2D byte array (height x width), or Nothing if conversion fails.

EXAMPLE CODE

```
let grayArray = ImgProc.Convert_RGBImage_To_GrayByteArray(bitmap);
```

32. ImgProc.Convert_RGBImage_To_GrayIntegerArray(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Integer(,)

DESCRIPTION

Converts RGB image pixels to grayscale integer array.

SYNTAX

```
let result = ImgProc.Convert_RGBImage_To_GrayIntegerArray(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to convert.	–
MaskImage Bitmap, Optional	Optional mask to constrain conversion.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Integer(,) — 2D integer array (height x width), or Nothing if conversion fails.

EXAMPLE CODE

```
let grayIntArray =
ImgProc.Convert_RGBImage_To_GrayIntegerArray(bitmap);
```

33. ImgProc.Convert_RGBImage_To_RGBByteArray(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Byte(,,)

DESCRIPTION

Extracts RGB image pixels into 3D byte array.

SYNTAX

```
let result = ImgProc.Convert_RGBImage_To_RGBByteArray(SourceImag
e, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to extract.	–
MaskImage Bitmap, Optional	Optional mask to constrain extraction.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Byte(,,) — 3D byte array (height x width x channels), or Nothing if extraction fails.

EXAMPLE CODE

```
let rgbArray = ImgProc.Convert_RGBImage_To_RGBByteArray(bitmap);
```

34. ImgProc.Count_ColorPixelOfInterest(SourceImage, ColorOfInterest_RGBInt, TotalPixelCount, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Integer

DESCRIPTION

Counts pixels of specific color.

SYNTAX

```
let result = ImgProc.Count_ColorPixelOfInterest(SourceImage, ColorOfInterest_RGBInt, TotalPixelCount, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to scan.	–
ColorOfInterest_RGBInt Integer	Target color as RGB integer.	–
TotalPixelCount Integer	Output total pixel count. Type: Integer; required; passed by reference.	–
MaskImage Bitmap, Optional	Optional mask to constrain counting.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Integer — Number of pixels matching the target color, or 0 if counting fails.

EXAMPLE CODE

```
let count = ImgProc.Count_ColorPixelOfInterest(bitmap, Color.Red.ToArgb(), totPix);
```

35. ImgProc.Count_ColorPixelOfInterest_FromGrayArray(ImageGrayArray(,), ColorOfInterestByte)

ColorOfInterestByte)

Returns: Integer

DESCRIPTION

Counts pixels of specified color in grayscale array.

SYNTAX

```
let result = ImgProc.Count_ColorPixelOfInterest_FromGrayArray(ImageGrayArray(,), ColorOfInterestByte);
```

PARAMETERS

Parameter	Description	Default
`ImageGrayArray(,)` Byte		–
`ColorOfInterestByte` Byte	Gray color value to count.	–

RETURN VALUE

Integer — Number of pixels matching color, or 0 if counting fails.

EXAMPLE CODE

```
let count =
ImgProc.Count_ColorPixelOfInterest_FromGrayArray(grayArray, 128);
```

36. ImgProc.Count_ColorPixelOfInterest_FromGrayArray_Region(ImageGrayArra ColorOfInterestByte, X1, Y1, X2, Y2)

Returns: Integer

DESCRIPTION

Counts pixels of specified color in region.

SYNTAX

```
let result = ImgProc.Count_ColorPixelOfInterest_FromGrayArray_Region(ImageGrayArray(,), ColorOfInterestByte, X1, Y1, X2, Y2);
```

PARAMETERS

Parameter	Description	Default
`ImageGrayArray(,)` Byte		–

ColorOfInterestByte Integer	Gray color value to count.	–
X1 Integer	Start X coordinate.	–
Y1 Integer	Start Y coordinate.	–
X2 Integer	End X coordinate.	–
Y2 Integer	End Y coordinate.	–

RETURN VALUE

`Integer` — Number of pixels matching color, or 0 if counting fails.

EXAMPLE CODE

```
let count =
ImgProc.Count_ColorPixelOfInterest_FromGrayArray_Region(grayArray,
128, 0, 0, 100, 100);
```

37. ImgProc.Count_NumberOfGrayPixel_OfInterest(SourceGrayImage, GrayColorOfInterest, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Integer

DESCRIPTION

Counts grayscale pixels of specific value.

SYNTAX

```
let result = ImgProc.Count_NumberOfGrayPixel_OfInterest(SourceGra
yImage, GrayColorOfInterest, MaskImage, MaskRegionBoundary, IsMas
kAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceGrayImage Bitmap	Source grayscale image.	–
GrayColorOfInterest	Target gray value to count.	–

Integer		
MaskImage Bitmap, Optional	Optional mask to constrain counting.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Integer — Number of grayscale pixels matching the target value, or 0 if counting fails.

EXAMPLE CODE

```
let count = ImgProc.Count_NumberOfGrayPixel_OfInterest(grayBitmap, 128);
```

38. ImgProc.CropExportRegions(SourceImage, MaskImage, use4WayConnectivity, outputModeType, marginPixels, transparentBackground, backgroundColorRed, backgroundColorGreen, backgroundColorBlue, destinationFolder, filePrefix, imageFormatType)

Returns: Integer

DESCRIPTION

Exports cropped regions based on a binary mask image directly from Bitmap objects.

SYNTAX

```
let result = ImgProc.CropExportRegions(SourceImage, MaskImage, use4WayConnectivity, outputModeType, marginPixels, transparentBackground, backgroundColorRed, backgroundColorGreen, backgroundColorBlue, destinationFolder, filePrefix, imageFormatType);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to crop from.	–
MaskImage Bitmap	Binary mask defining regions.	–

use4WayConnectivity Boolean, Optional	Whether to use 4-way connectivity.	False
outputModeType Integer, Optional	Integer representing the region export output mode. Supported outputModeType values: 0: ROI - rectangular region of interest 1: ExactShape - exact region shape 2: Binary - binary mask	1
marginPixels Integer, Optional	Margin to add around regions.	0
transparentBackground Boolean, Optional	Whether to use transparent background.	True
backgroundColorRed Integer, Optional	Background red component (0-255).	0
backgroundColorGreen Integer, Optional	Background green component (0-255).	0
backgroundColorBlue Integer, Optional	Background blue component (0-255).	0
destinationFolder String, Optional	Folder to save exported regions.	"C:₩₩ExportedRegions"
filePrefix String, Optional	Prefix for exported file names.	"Region₩_"
imageFormatType Integer, Optional	Integer representing the export image format. Supported imageFormatType values: 0: PNG 1: JPEG 2: BMP	0

RETURN VALUE

`Integer` — Number of successfully exported regions.

EXAMPLE CODE

```
let exported = ImgProc.CropExportRegions(bitmap, maskBitmap, false, 1, 0, true);
```

39. ImgProc.Distance(X1, Y1, X2, Y2) Returns: Double

DESCRIPTION

Calculates Euclidean distance between two points.

SYNTAX

```
let result = ImgProc.Distance(X1, Y1, X2, Y2);
```

PARAMETERS

Parameter	Description	Default
X1 Double	X-coordinate of first point.	–
Y1 Double	Y-coordinate of first point.	–
X2 Double	X-coordinate of second point.	–
Y2 Double	Y-coordinate of second point.	–

RETURN VALUE

`Double` — Euclidean distance between the two points.

EXAMPLE CODE

```
let dist = ImgProc.Distance(0, 0, 3, 4);
```

40. ImgProc.Draw_Histogram(ProfileData, TotPixelCount, Pic_Target, ProfileColor)

Returns: Bitmap

DESCRIPTION

Draws histogram within picture control.

SYNTAX

```
let result = ImgProc.Draw_Histogram(ProfileData, TotPixelCount, Pic_Target, ProfileColor);
```

PARAMETERS

Parameter	Description	Default
ProfileData Single()	Array of histogram bin data.	–
TotPixelCount Integer	Total pixel count for normalization.	–
Pic_Target PictureBox	Target picture control to draw in. Type: PictureBox; required; passed by reference.	–
ProfileColor Color	Color for histogram bars.	–

RETURN VALUE

`Bitmap` — Bitmap containing the histogram visualization, or Nothing if drawing fails.

EXAMPLE CODE

```
let hist = ImgProc.Draw_Histogram(data, 1000000, picTarget, Color.Blue);
```

41. ImgProc.Draw_LineProfile(ProfileData, Pic_Target, ProfileColor, IsCenterLined)

Returns: Bitmap

DESCRIPTION

Draws line profile within picture control.

SYNTAX

```
let result = ImgProc.Draw_LineProfile(ProfileData, Pic_Target, ProfileColor, IsCenterLined);
```

PARAMETERS

Parameter	Description	Default
ProfileData Single()	Array of profile data values.	–
Pic_Target PictureBox	Target picture control to draw in. Type: PictureBox; required; passed by reference.	–
ProfileColor Color	Color for profile line.	–
IsCenterLined Boolean, Optional	Whether to draw center line.	False

RETURN VALUE

Bitmap — Bitmap containing the profile visualization, or Nothing if drawing fails.

EXAMPLE CODE

```
let profile = ImgProc.Draw_LineProfile(data, picTarget, Color.Red);
```

42. ImgProc.Extract_PixelLocations(SourceGrayImage, GrayColorOfInterest, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Point()

DESCRIPTION

Extracts pixel coordinates of specified gray color.

SYNTAX

```
let result = ImgProc.Extract_PixelLocations(SourceGrayImage, GrayColorOfInterest, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceGrayImage Bitmap	Source grayscale image.	–
GrayColorOfInterest Integer	Gray color value to find.	–
MaskImage	Optional mask to constrain	Nothing

Bitmap, Optional	extraction.	
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Point() — Array of Point coordinates matching color, or Nothing if extraction fails.

EXAMPLE CODE

```
let points = ImgProc.Extract_PixelLocations(grayBitmap, 128);
```

43. ImgProc.Extract_PixelLocations_FromByte(SrcByteArray, GrayColorOfInterest, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Point()

DESCRIPTION

Extracts pixel coordinates from byte array.

SYNTAX

```
let result = ImgProc.Extract_PixelLocations_FromByte(SrcByteArray, GrayColorOfInterest, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SrcByteArray Byte(,)	Source byte array [height, width].	–
GrayColorOfInterest Integer	Gray color value to find.	–
MaskImage Bitmap, Optional	Optional mask to constrain extraction.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected	Whether entire region is	True

Boolean, Optional selected.

RETURN VALUE

Point() — Array of Point coordinates matching color, or Nothing if extraction fails.

EXAMPLE CODE

```
let points = ImgProc.Extract_PixelLocations_FromByte(byteData, 128);
```

44. ImgProc.Extract_SkeletonPoints(SkeletonArray(,), foreground)

Returns: Point()

DESCRIPTION

Extracts all skeleton points from array.

SYNTAX

```
let result = ImgProc.Extract_SkeletonPoints(SkeletonArray(,), foreground);
```

PARAMETERS

Parameter	Description	Default
`SkeletonArray(,)` Byte		–
`foreground` Byte	Foreground byte value.	–

RETURN VALUE

Point() — Array of skeleton points, or Nothing if extraction fails.

EXAMPLE CODE

```
let points = ImgProc.Extract_SkeletonPoints(skelArray, 255);
```

45. ImgProc.Get_Avgerage_Color_3x3Box_ExceptOneColor(TargetX, TargetY, TargetColor_R, TargetColor_G, TargetColor_B, ColorChannelNum)

Returns: Byte

DESCRIPTION

Gets average color from 3x3 box excluding target color.

SYNTAX

```
let result = ImgProc.Get_Avgerage_Color_3x3Box_ExceptOneColor(TargetX, TargetY, TargetColor_R, TargetColor_G, TargetColor_B, ColorChannelNum);
```

PARAMETERS

Parameter	Description	Default
`TargetX` Integer	X coordinate of center pixel.	–
`TargetY` Integer	Y coordinate of center pixel.	–
`TargetColor_R` Byte	Red channel of color to exclude.	–
`TargetColor_G` Byte	Green channel of color to exclude.	–
`TargetColor_B` Byte	Blue channel of color to exclude.	–
`ColorChannelNum` Integer	Integer representing the color channel. Supported ColorChannelNum values: 0: Red 1: Green 2: Blue	–

RETURN VALUE

`Byte` — Average color value for the channel, or 0 if operation fails.

EXAMPLE CODE

```
let avgRed = ImgProc.Get_Avgerage_Color_3x3Box_ExceptOneColor(10, 10, 0, 0, 0, 0);
```

46. ImgProc.Get_ColorRGB2DMap(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Object

DESCRIPTION

Gets 2D color RGB map from image.

SYNTAX

```
let result = ImgProc.Get_ColorRGB2DMap(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to extract map from.	–
MaskImage Bitmap, Optional	Optional mask to constrain extraction.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Object — 2D object array containing color RGB map, or Nothing if extraction fails.

EXAMPLE CODE

```
let rgbMap = ImgProc.Get_ColorRGB2DMap(bitmap);
```

47. ImgProc.Get_GrayValues(SourceImage, x1, y1, x2, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Byte()

DESCRIPTION

Extracts grayscale values from image line.

SYNTAX

```
let result = ImgProc.Get_GrayValues(SourceImage, x1, y1, x2, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to extract from.	–
x1	Start X coordinate	–

x1 Integer	Start X coordinate.	
y1 Integer	Start Y coordinate.	–
x2 Integer	End X coordinate.	–
MaskImage Bitmap, Optional	Optional mask to constrain extraction.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Byte()` — Array of grayscale values, or Nothing if extraction fails.

EXAMPLE CODE

```
let grayValues = ImgProc.Get_GrayValues(bitmap, 0, 0, 100);
```

48. ImgProc.Get_Histogram(SourceImage, TotalPixelCount, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Single(,)

DESCRIPTION

Calculates histogram of image pixel intensities.

SYNTAX

```
let result = ImgProc.Get_Histogram(SourceImage, TotalPixelCount, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to analyze.	–
TotalPixelCount Integer	Output total pixel count. Type: Integer; required; passed by reference.	–

MaskImage Bitmap, Optional	Optional mask to constrain histogram.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Single(,) — 2D array of histogram data, or Nothing if calculation fails.

EXAMPLE CODE

```
let hist = ImgProc.Get_Histogram(bitmap, totalPix);
```

49. ImgProc.Get_LineProfile(SourceImage, StartAt, EndAt, LineThickness, IsNormalizeBy255)

Returns: Single(,)

DESCRIPTION

Gets pixel intensity profile along a line.

SYNTAX

```
let result = ImgProc.Get_LineProfile(SourceImage, StartAt, EndAt, LineThickness, IsNormalizeBy255);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to extract profile from.	–
StartAt Point	Starting point of profile line.	–
EndAt Point	Ending point of profile line.	–
LineThickness Integer	Thickness of the profile line.	–
IsNormalizeBy255 Boolean, Optional	Whether to normalize values by 255.	False

RETURN VALUE

`Single(,)` — 2D array of profile data, or Nothing if extraction fails.

EXAMPLE CODE

```
let profile = ImgProc.Get_LineProfile(bitmap, New Point(0, 0), New Point(100, 0), 5);
```

50. ImgProc.Get_MaxValue(Value1, Value2, Value3, Value4, Value5)

Returns: Single

DESCRIPTION

Returns maximum value among five values.

SYNTAX

```
let result = ImgProc.Get_MaxValue(Value1, Value2, Value3, Value4, Value5);
```

PARAMETERS

Parameter	Description	Default
Value1 Single	First value.	–
Value2 Single	Second value.	–
Value3 Single	Third value.	–
Value4 Single	Fourth value.	–
Value5 Single	Fifth value.	–

RETURN VALUE

`Single` — Maximum among the five values.

EXAMPLE CODE

```
let maxVal = ImgProc.Get_MaxValue(1.0, 2.5, 0.5, 3.0, 1.5);
```

51. ImgProc.Get_MinValue(Value1, Value2, Value3, Value4, Value5)

51. ImgProc.Get_MinValue(Value1, Value2, Value3, Value4, Value5)

Returns: Single

DESCRIPTION

Returns minimum value among five values.

SYNTAX

```
let result = ImgProc.Get_MinValue(Value1, Value2, Value3, Value4, Value5);
```

PARAMETERS

Parameter	Description	Default
`Value1` Single	First value.	–
`Value2` Single	Second value.	–
`Value3` Single	Third value.	–
`Value4` Single	Fourth value.	–
`Value5` Single	Fifth value.	–

RETURN VALUE

`Single` — Minimum among the five values.

EXAMPLE CODE

```
let minVal = ImgProc.Get_MinValue(1.0, 2.5, 0.5, 3.0, 1.5);
```

52. ImgProc.Get_MotionDetectHits(SourceImage1, SourceImage2, StepSize, Tolerance)

Returns: Single

DESCRIPTION

Detects motion between two images.

SYNTAX

```
let result = ImgProc.Get_MotionDetectHits(SourceImage1, SourceImage2, StepSize, Tolerance);
```

PARAMETERS

Parameter	Description	Default
`SourceImage1` Bitmap	First source image.	–
`SourceImage2` Bitmap	Second source image to compare.	–
`StepSize` Single	Step size for motion sampling.	–
`Tolerance` Single	Motion detection tolerance.	–

RETURN VALUE

`Single` — Measure of motion detected between images.

EXAMPLE CODE

```
let motion = ImgProc.Get_MotionDetectHits(frame1, frame2, 1.0,
10.0);
```

53. ImgProc.Get_PixelColor(SourceBitmap, TargetX, TargetY)

Returns: Integer

DESCRIPTION

Gets pixel color from bitmap at specified coordinates.

SYNTAX

```
let result = ImgProc.Get_PixelColor(SourceBitmap, TargetX, Target
Y);
```

PARAMETERS

Parameter	Description	Default
`SourceBitmap` Bitmap	Source bitmap to sample.	–
`TargetX` Integer	X-coordinate within bitmap.	–
`TargetY`	Y-coordinate within bitmap.	–

Integer

RETURN VALUE

Integer — Integer color value at specified bitmap location, or 0 if retrieval fails.

EXAMPLE CODE

```
let pixelColor = ImgProc.Get_PixelColor(bitmap, 50, 75);
```

54. ImgProc.Get_PixelColor_From1PixelInScreen(TargetX, TargetY)

Returns: Integer

DESCRIPTION

Gets pixel color from screen at specified coordinates.

SYNTAX

```
let result = ImgProc.Get_PixelColor_From1PixelInScreen(TargetX, TargetY);
```

PARAMETERS

Parameter	Description	Default
TargetX Integer	X-coordinate on screen.	–
TargetY Integer	Y-coordinate on screen.	–

RETURN VALUE

Integer — Integer color value at specified screen location, or 0 if retrieval fails.

EXAMPLE CODE

```
let pixelColor = ImgProc.Get_PixelColor_From1PixelInScreen(100, 100);
```

55. ImgProc.Get_PixelColor_FromScreen(TargetX, TargetY)

Returns: Integer

DESCRIPTION

Gets pixel color from screen.

SYNTAX

```
let result = ImgProc.Get_PixelColor_FromScreen(TargetX, TargetY);
```

PARAMETERS

Parameter	Description	Default
TargetX Integer	X-coordinate on screen.	–
TargetY Integer	Y-coordinate on screen.	–

RETURN VALUE

Integer — Integer color value at specified screen location, or 0 if retrieval fails.

EXAMPLE CODE

```
let pixelColor = ImgProc.Get_PixelColor_FromScreen(200, 150);
```

56. ImgProc.Get_PixelColorValues(SourceImage, X1, Y1, ReturnRed, ReturnGreen, ReturnBlue, ReturnGray, ReturnHue, ReturnSat, ReturnIntensity, ReturnCyan, ReturnMagenta, ReturnYellow, ReturnK, ReturnLab_L, ReturnLab_a, ReturnLab_b, ReturnRG_Red, ReturnRG_Green, ReturnRG_Blue)

Returns: Void

DESCRIPTION

Gets individual pixel color values in multiple color spaces.

SYNTAX

```
ImgProc.Get_PixelColorValues(SourceImage, X1, Y1, ReturnRed, ReturnGreen, ReturnBlue, ReturnGray, ReturnHue, ReturnSat, ReturnIntensity, ReturnCyan, ReturnMagenta, ReturnYellow, ReturnK, ReturnLab_L, ReturnLab_a, ReturnLab_b, ReturnRG_Red, ReturnRG_Green, ReturnRG_Blue);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to sample.	–
X1 Integer	X-coordinate of pixel.	–

Y1 Integer	Y-coordinate of pixel.	–
ReturnRed Byte	Output red component. Type: Byte; required; passed by reference.	–
ReturnGreen Byte	Output green component. Type: Byte; required; passed by reference.	–
ReturnBlue Byte	Output blue component. Type: Byte; required; passed by reference.	–
ReturnGray Byte	Output gray value. Type: Byte; required; passed by reference.	–
ReturnHue Single	Output hue. Type: Single; required; passed by reference.	–
ReturnSat Single	Output saturation. Type: Single; required; passed by reference.	–
ReturnIntensity Single	Output intensity. Type: Single; required; passed by reference.	–
ReturnCyan Single	Output cyan. Type: Single; required; passed by reference.	–
ReturnMagenta Single	Output magenta. Type: Single; required; passed by reference.	–
ReturnYellow Single	Output yellow. Type: Single; required; passed by reference.	–
ReturnK Single	Output K (black) in CMYK. Type: Single; required; passed by reference.	–
ReturnLab_L Single	Output Lab L. Type: Single; required; passed by reference.	–

ReturnLab_a Single	Output Lab a. Type: Single; required; passed by reference.	–
ReturnLab_b Single	Output Lab b. Type: Single; required; passed by reference.	–
ReturnRG_Red Single	Output rg chromaticity red. Type: Single; required; passed by reference.	–
ReturnRG_Green Single	Output rg chromaticity green. Type: Single; required; passed by reference.	–
ReturnRG_Blue Single	Output rg chromaticity blue. Type: Single; required; passed by reference.	–

RETURN VALUE

Void

EXAMPLE CODE

```
ImgProc.Get_PixelColorValues(bitmap, 50, 50, r, g, b, gray, hue,
sat, int, cyan, mag, yel, k, labL, labA, labB, rgR, rgG, rgB);
```

57. ImgProc.Get_PixelLocationArray(SourceGrayImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Point()

DESCRIPTION

Gets all white pixel locations in grayscale image.

SYNTAX

```
let result = ImgProc.Get_PixelLocationArray(SourceGrayImage, Mask
Image, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceGrayImage Bitmap	Source grayscale image to analyze.	–

MaskImage Bitmap, Optional	Optional mask to constrain search.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Point() — Array of Point coordinates for white pixels, or Nothing if search fails.

EXAMPLE CODE

```
let whitePoints = ImgProc.Get_PixelLocationArray(bitmap);
```

58. ImgProc.Get_PixelLocationArray_Bk(SourceGrayImage, DestPixelGray, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Point()

DESCRIPTION

Gets pixel locations matching specified grayscale value.

SYNTAX

```
let result = ImgProc.Get_PixelLocationArray_Bk(SourceGrayImage, DestPixelGray, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceGrayImage Bitmap	Source grayscale image.	–
DestPixelGray Byte	Target grayscale value to find (0-255).	–
MaskImage Bitmap, Optional	Optional mask to constrain search.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected	Whether entire region is	True

Boolean, Optional	selected.

RETURN VALUE

`Point()` — Array of Point coordinates, or Nothing if search fails.

EXAMPLE CODE

```
let points = ImgProc.Get_PixelLocationArray_Bk(bitmap, 128);
```

59. ImgProc.Get_TrueSkeletonLength(SkeletonArray(,), foreground, umPerPixels, SplineVertexCount)

Returns: Single

DESCRIPTION

Calculates true skeleton curve length with spline.

SYNTAX

```
let result = ImgProc.Get_TrueSkeletonLength(SkeletonArray(,), foreground, umPerPixels, SplineVertexCount);
```

PARAMETERS

Parameter	Description	Default
SkeletonArray(,) Byte		–
foreground Byte	Foreground byte value.	–
umPerPixels Single	Micrometer per pixel ratio.	–
SplineVertexCount Integer	Number of spline vertices.	–

RETURN VALUE

`Single` — True skeleton length, or 0 if calculation fails.

EXAMPLE CODE

```
let length = ImgProc.Get_TrueSkeletonLength(skelArray, 255, 1.0, 100);
```

60. ImgProc.GrayScaling_PutTo_2DByte(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Byte(,)

DESCRIPTION

Converts image to grayscale and stores in 2D byte array.

SYNTAX

```
let result = ImgProc.GrayScaling_PutTo_2DByte(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to convert.	–
MaskImage Bitmap, Optional	Optional mask to constrain conversion.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Byte(,) — 2D byte array (height x width), or Nothing if conversion fails.

EXAMPLE CODE

```
let grayArray = ImgProc.GrayScaling_PutTo_2DByte(bitmap);
```

61. ImgProc.GrayScaling_PutTo_2DInt(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Integer(,)

DESCRIPTION

Converts image to grayscale and stores in 2D integer array.

SYNTAX

```
let result = ImgProc.GrayScaling_PutTo_2DInt(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to convert.	–
MaskImage Bitmap, Optional	Optional mask to constrain conversion.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Integer(,) — 2D integer array (height x width), or Nothing if conversion fails.

EXAMPLE CODE

```
let grayIntArray = ImgProc.GrayScaling_PutTo_2DInt(bitmap);
```

62. ImgProc.GrayScaling_ReturnImg_From_2DInt(SourceImage, Gray2DInt, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Converts image to grayscale and returns result from 2D integer array.

SYNTAX

```
let result = ImgProc.GrayScaling_ReturnImg_From_2DInt(SourceImage, Gray2DInt, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to process.	–
Gray2DInt Integer(,)	2D integer array (height x width) to use.	–
MaskImage	Optional mask to constrain	Nothing

Bitmap, Optional	processing.	
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Processed image, or Nothing if conversion fails.

EXAMPLE CODE

```
let result = ImgProc.GrayScaling_ReturnImg_From_2DInt(bitmap, intArray);
```

63. ImgProc.IsSameImage(IsTheFirstImage, SourceImage)

Returns: Boolean

DESCRIPTION

Checks if image is same as stored reference.

SYNTAX

```
let result = ImgProc.IsSameImage(IsTheFirstImage, SourceImage);
```

PARAMETERS

Parameter	Description	Default
IsTheFirstImage Boolean	Whether to check against first or second reference.	–
SourceImage Bitmap	Source image to check.	–

RETURN VALUE

Boolean — True if image matches reference, False otherwise.

EXAMPLE CODE

```
let isSame = ImgProc.IsSameImage(true, bitmap);
```

64. ImgProc.IsSameImage_QuickCheck(IsTheFirstImage, SourceImage, x1, y1, x2)

Returns: Boolean

DESCRIPTION

Quick check if image is same as stored reference.

SYNTAX

```
let result = ImgProc.IsSameImage_QuickCheck(IsTheFirstImage, SourceImage, x1, y1, x2);
```

PARAMETERS

Parameter	Description	Default
`IsTheFirstImage` Boolean	Whether to check against first or second reference.	–
`SourceImage` Bitmap	Source image to check.	–
`x1` Integer	Start X coordinate.	–
`y1` Integer	Start Y coordinate.	–
`x2` Integer	End X coordinate.	–

RETURN VALUE

Boolean — True if image matches reference, False otherwise.

EXAMPLE CODE

```
let isSame = ImgProc.IsSameImage_QuickCheck(true, bitmap, 0, 0, 100);
```

65. ImgProc.Overlay_GrayIntArray_On_Image(SourceImage, Gray2DInt, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Overlays grayscale integer array values as colors on image.

SYNTAX

```
let result = ImgProc.Overlay_GrayIntArray_On_Image(SourceImage, Gray2DInt, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
`SourceImage` Bitmap	Source image to overlay on.	–
`Gray2DInt` Integer(,)	2D integer array (height x width) values.	–
`MaskImage` Bitmap	Mask image.	–
`MaskRegionBoundary` Rectangle	Boundary rectangle for masked region.	–
`IsMaskAllRegionSelected` Boolean	Whether entire region is selected.	–

RETURN VALUE

`Bitmap` — Image with overlay applied, or Nothing if overlay fails.

EXAMPLE CODE

```
let overlaid = ImgProc.Overlay_GrayIntArray_On_Image(bitmap, intArray, mask, rect, true);
```

66. ImgProc.Quantify_ColorPixel(SourceImage, Return_ColorTable(,), TotalPixelCount, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Integer

DESCRIPTION

Counts and classifies all color pixels.

SYNTAX

```
let result = ImgProc.Quantify_ColorPixel(SourceImage, Return_ColorTable(,), TotalPixelCount, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default

SourceImage Bitmap	Source image to analyze.	–
Return_ColorTable(,) Integer		–
TotalPixelCount Integer	Output total pixel count. Type: Integer; required; passed by reference.	–
MaskImage Bitmap, Optional	Optional mask to constrain analysis.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Integer — Number of color classifications found, or 0 if quantification fails.

EXAMPLE CODE

```
let colorCount = ImgProc.Quantify_ColorPixel(bitmap, colorTbl,
totPix);
```

67. ImgProc.RegionExtract_BasicAnalysis(Src_IDMap(,), Src_ColorTable(,))

Returns: Double(,)

DESCRIPTION

Extracts basic region analysis statistics from label and color maps.

SYNTAX

```
let result = ImgProc.RegionExtract_BasicAnalysis(Src_IDMap(,), Sr
c_ColorTable(,));
```

PARAMETERS

Parameter	Description	Default
Src_IDMap(,) Integer		–

Src_ColorTable(,)

Byte

–

RETURN VALUE

Double(,) — 2D array of analysis statistics, or Nothing if analysis fails.

EXAMPLE CODE

```
let stats = ImgProc.RegionExtract_BasicAnalysis(labelMap,
colorTable);
```

68. ImgProc.RegionExtract_Convert_LabelIDMap_To_Image(LabelIDMap, ObjectRed, ObjectGreen, ObjectBlue)

Returns: Bitmap

DESCRIPTION

Converts label ID map to colored image representation.

SYNTAX

```
let result = ImgProc.RegionExtract_Convert_LabelIDMap_To_Image(La
belIDMap, ObjectRed, ObjectGreen, ObjectBlue);
```

PARAMETERS

Parameter	Description	Default
LabelIDMap Integer(,)	Region label map.	–
ObjectRed Byte, Optional	Red channel value for objects (0-255).	255
ObjectGreen Byte, Optional	Green channel value for objects (0-255).	255
ObjectBlue Byte, Optional	Blue channel value for objects (0-255).	255

RETURN VALUE

Bitmap — Image with regions colored, or Nothing if conversion fails.

EXAMPLE CODE

```
let colored =
ImgProc.RegionExtract_Convert_LabelIDMap_To_Image(labelMap, 255, 0,
0);
```

69. ImgProc.RegionExtract_Create_GreenObjectOutlineImage(SourceLabelIDMap SourceIDINfoIndex, x1, x2, y1, y2, DilateBoxSize)

Returns: Bitmap

DESCRIPTION

Creates green outlined image from label map with optional dilation.

SYNTAX

```
let result = ImgProc.RegionExtract_Create_GreenObjectOutlineImage(SourceLabelIDMap(,), SourceIDINfoIndex, x1, x2, y1, y2, DilateBoxSize);
```

PARAMETERS

Parameter	Description	Default
SourceLabelIDMap(,) Integer		–
SourceIDINfoIndex Integer	Region ID index to extract.	–
x1 Integer	Left boundary coordinate.	–
x2 Integer	Right boundary coordinate.	–
y1 Integer	Top boundary coordinate.	–
y2 Integer	Bottom boundary coordinate.	–
DilateBoxSize Integer	Size of dilation box kernel.	–

RETURN VALUE

`Bitmap` — Green outlined image, or Nothing if creation fails.

EXAMPLE CODE

```
let outline =
ImgProc.RegionExtract_Create_GreenObjectOutlineImage(labelMap, 1,
```

```
0, 100, 0, 100, 3);
```

70. ImgProc.RegionExtract_Create_OutlinedOriginalImage(SourceImage, LabelIDMap, ObjectRed, ObjectGreen, ObjectBlue)

Returns: Bitmap

DESCRIPTION

Creates image with region outlines overlaid on original image.

SYNTAX

```
let result = ImgProc.RegionExtract_Create_OutlinedOriginalImage(SourceImage, LabelIDMap, ObjectRed, ObjectGreen, ObjectBlue);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to overlay on.	–
LabelIDMap Integer(,)	Region label map.	–
ObjectRed Byte, Optional	Red channel value for outlines (0-255).	255
ObjectGreen Byte, Optional	Green channel value for outlines (0-255).	255
ObjectBlue Byte, Optional	Blue channel value for outlines (0-255).	255

RETURN VALUE

`Bitmap` — Image with outlines overlaid, or Nothing if creation fails.

EXAMPLE CODE

```
let outlined = ImgProc.RegionExtract_Create_OutlinedOriginalImage(bitmap, labelMap);
```

71. ImgProc.RegionExtract_Create_WhiteObjectOutlineImage(SourceLabelIDMap

SourceIDINfoIndex, x1, x2, y1, y2, DilateBoxSize)

Returns: Bitmap

DESCRIPTION

Creates white outlined image from label map with optional dilation.

SYNTAX

```
let result = ImgProc.RegionExtract_Create_WhiteObjectOutlineImage(SourceLabelIDMap(,), SourceIDINfoIndex, x1, x2, y1, y2, DilateBoxSize);
```

PARAMETERS

Parameter	Description	Default
`SourceLabelIDMap(,)` Integer		–
`SourceIDINfoIndex` Integer	Region ID index to extract.	–
`x1` Integer	Left boundary coordinate.	–
`x2` Integer	Right boundary coordinate.	–
`y1` Integer	Top boundary coordinate.	–
`y2` Integer	Bottom boundary coordinate.	–
`DilateBoxSize` Integer	Size of dilation box kernel.	–

RETURN VALUE

`Bitmap` — White outlined image, or Nothing if creation fails.

EXAMPLE CODE

```
let outline = ImgProc.RegionExtract_Create_WhiteObjectOutlineImage(labelMap, 1, 0, 100, 0, 100, 3);
```

72. ImgProc.RegionExtract_Crop_LabelIDMap_To_2DByte(LabelIDMap, IDInfoIndex, X1, X2, Y1, Y2)

Returns: Byte(,)

DESCRIPTION

Extracts and converts specified regions from label map to byte array.

SYNTAX

```
let result = ImgProc.RegionExtract_Crop_LabelIDMap_To_2DByte(LabelIDMap, IDInfoIndex, X1, X2, Y1, Y2);
```

PARAMETERS

Parameter	Description	Default
`LabelIDMap` Integer(,)	Region label map.	–
`IDInfoIndex` Integer	Region ID index to extract.	–
`X1` Integer	Left boundary coordinate.	–
`X2` Integer	Right boundary coordinate.	–
`Y1` Integer	Top boundary coordinate.	–
`Y2` Integer	Bottom boundary coordinate.	–

RETURN VALUE

`Byte(,)` — 2D byte array of extracted region, or Nothing if extraction fails.

EXAMPLE CODE

```
let regionBytes = ImgProc.RegionExtract_Crop_LabelIDMap_To_2DByte(labelMap, 1, 0, 100, 0, 100);
```

73. ImgProc.RegionExtract_Crop_WhiteObjectImage(SourceLabelIDMap(,), SourceIDINfoIndex, x1, x2, y1, y2)

Returns: Bitmap

DESCRIPTION

Crops white object region from label ID map.

SYNTAX

```
let result = ImgProc.RegionExtract_Crop_WhiteObjectImage(SourceLabelIDMap(,), SourceIDINfoIndex, x1, x2, y1, y2);
```

PARAMETERS

Parameter	Description	Default
`SourceLabelIDMap(,)` Integer		–
`SourceIDINfoIndex` Integer	Region ID index to extract.	–
`x1` Integer	Left boundary coordinate.	–
`x2` Integer	Right boundary coordinate.	–
`y1` Integer	Top boundary coordinate.	–
`y2` Integer	Bottom boundary coordinate.	–

RETURN VALUE

`Bitmap` — White cropped region image, or Nothing if cropping fails.

EXAMPLE CODE

```
let cropped = ImgProc.RegionExtract_Crop_WhiteObjectImage(labelMap, 1, 0, 100, 0, 100);
```

74. ImgProc.RegionExtract_Update_LabelIDMap_ByIncluding(LabelIDMap, IDArray)

Returns: Integer(,)

DESCRIPTION

Updates label ID map to include only specified region IDs.

SYNTAX

```
let result = ImgProc.RegionExtract_Update_LabelIDMap_ByIncluding
(LabelIDMap, IDArray);
```

PARAMETERS

Parameter	Description	Default
LabelIDMap Integer(,)	Region label map to update.	–
IDArray Dictionary(Of Integer, Boolean)	Dictionary of region IDs to include.	–

RETURN VALUE

Integer(,) — Updated label ID map, or Nothing if update fails.

EXAMPLE CODE

```
let filtered =
ImgProc.RegionExtract_Update_LabelIDMap_ByIncluding(labelMap,
idDict);
```

75. ImgProc.RegionExtract_Update_LabelIDMap_To_Image(SourceImage, LabelIDMap, RegionColorTable, IsOverlap, BkColor, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Converts label ID map values to corresponding image colors.

SYNTAX

```
let result = ImgProc.RegionExtract_Update_LabelIDMap_To_Image(Sou
rceImage, LabelIDMap, RegionColorTable, IsOverlap, BkColor, MaskI
mage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to use for color reference.	–
LabelIDMap Integer(,)	Region label map.	–

RegionColorTable Byte(,)	Region color table.	–
IsOverlap Boolean, Optional	Whether to overlap with source image.	True
BkColor Integer, Optional	Background color to use.	0
MaskImage Bitmap, Optional	Optional mask to constrain conversion.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Image with labels converted to colors, or Nothing if conversion fails.

EXAMPLE CODE

```
let coloredRegions =
ImgProc.RegionExtract_Update_LabelIDMap_To_Image(bitmap, labelMap,
colorTable);
```

Image Transforms & Filters 80 functions

1. ImgProc.AdaptiveBinarization(SourceImage, LocalBoxSizeValue, IsObjectWhite, ThresholdLevel, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Converts to binary using local adaptive threshold.

SYNTAX

```
let result = ImgProc.AdaptiveBinarization(SourceImage, LocalBoxSi
zeValue, IsObjectWhite, ThresholdLevel, IsShowProgress, MaskImag
e, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to binarize.	–
LocalBoxSizeValue Integer	Size of local adaptive box.	–
IsObjectWhite Boolean	Whether object is white (true) or black (false).	–
ThresholdLevel Integer	Threshold level for binarization.	–
IsShowProgress Boolean, Optional	Whether to show progress.	`True`
MaskImage Bitmap, Optional	Optional mask to constrain binarization.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Bitmap` — Binarized image, or Nothing if binarization fails.

EXAMPLE CODE

```
let binary = ImgProc.AdaptiveBinarization(bitmap, 15, true, 10);
```

2. ImgProc.AdaptiveContrast(SourceImage, Strength, TilesX, TilesY, ClipLimit, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Applies adaptive contrast enhancement with histogram equalization.

SYNTAX

```
let result = ImgProc.AdaptiveContrast(SourceImage, Strength, TilesX, TilesY, ClipLimit, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to enhance.	–
Strength Single, Optional	Enhancement strength (0.0-1.0).	0.6F
TilesX Integer, Optional	Number of horizontal tiles.	8
TilesY Integer, Optional	Number of vertical tiles.	8
ClipLimit Single, Optional	Histogram clip limit.	1.5F
IsShowProgress Boolean, Optional	Whether to show progress.	True
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional		True

RETURN VALUE

Bitmap — Contrast-enhanced image, or Nothing if enhancement fails.

EXAMPLE CODE

```
let enhanced = ImgProc.AdaptiveContrast(bitmap, 0.6, 8, 8, 1.5);
```

3. ImgProc.AddMargin(SourceImage, LeftMargin, RightMargin, TopMargin, BottomMargin)

Returns: Bitmap

DESCRIPTION

Adds margins to image edges with background color.

SYNTAX

```
let result = ImgProc.AddMargin(SourceImage, LeftMargin, RightMarg
```

```
in, TopMargin, BottomMargin);
```

PARAMETERS

Parameter	Description	Default
`SourceImage` Bitmap	Source image to add margins to.	–
`LeftMargin` Integer	Left margin size in pixels.	–
`RightMargin` Integer	Right margin size in pixels.	–
`TopMargin` Integer	Top margin size in pixels.	–
`BottomMargin` Integer	Bottom margin size in pixels.	–

RETURN VALUE

`Bitmap` — Image with added margins, or Nothing if operation fails.

EXAMPLE CODE

```
let margined = ImgProc.AddMargin(bitmap, 10, 10, 10, 10);
```

4. ImgProc.AutoContrast(SourceImage, ClipPercent) Returns: Bitmap

DESCRIPTION

Auto adjusts contrast automatically.

SYNTAX

```
let result = ImgProc.AutoContrast(SourceImage, ClipPercent);
```

PARAMETERS

Parameter	Description	Default
`SourceImage` Bitmap	Source image to adjust.	–
`ClipPercent` Single, Optional	Clip percentage for contrast range.	`0.5F`

RETURN VALUE

`Bitmap` — Contrast-adjusted image, or Nothing if adjustment fails.

EXAMPLE CODE

```
let contrasted = ImgProc.AutoContrast(bitmap, 0.5);
```

5. ImgProc.AutoEqualize(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Auto equalizes histogram.

SYNTAX

```
let result = ImgProc.AutoEqualize(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to equalize.	–
MaskImage Bitmap, Optional	Optional mask to constrain operation.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Bitmap` — Histogram-equalized image, or Nothing if equalization fails.

EXAMPLE CODE

```
let equalized = ImgProc.AutoEqualize(bitmap);
```

6. ImgProc.AutoGamma(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Auto adjusts gamma curve.

SYNTAX

```
let result = ImgProc.AutoGamma(SourceImage, MaskImage, MaskRegion
Boundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to adjust.	–
MaskImage Bitmap, Optional	Optional mask to constrain operation.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Bitmap` — Gamma-adjusted image, or Nothing if adjustment fails.

EXAMPLE CODE

```
let gammaed = ImgProc.AutoGamma(bitmap);
```

7. ImgProc.AutoTone(SourceImage, ClipPercent, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Auto adjusts tonal range.

SYNTAX

```
let result = ImgProc.AutoTone(SourceImage, ClipPercent, MaskImag
e, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage	Source image to adjust	–

Bitmap		
ClipPercent Single, Optional	Clip percentage for tonal range.	`0.5F`
MaskImage Bitmap, Optional	Optional mask to constrain operation.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Bitmap` — Tone-adjusted image, or Nothing if adjustment fails.

EXAMPLE CODE

```
let toned = ImgProc.AutoTone(bitmap, 0.5);
```

8. ImgProc.AutoWhiteBalance(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Auto adjusts white balance.

SYNTAX

```
let result = ImgProc.AutoWhiteBalance(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to adjust.	–
MaskImage Bitmap, Optional	Optional mask to constrain operation.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected	Whether entire region is	`True`

Boolean, Optional	selected.

RETURN VALUE

`Bitmap` — White-balanced image, or Nothing if adjustment fails.

EXAMPLE CODE

```
let balanced = ImgProc.AutoWhiteBalance(bitmap);
```

9. ImgProc.BoxAveraging(SourceImage, BoxSizeHalf, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Applies box averaging (simple blur) filter.

SYNTAX

```
let result = ImgProc.BoxAveraging(SourceImage, BoxSizeHalf, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to blur.	–
BoxSizeHalf Integer	Half-size of averaging box kernel.	–
MaskImage Bitmap, Optional	Optional mask to constrain blur.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Bitmap` — Blurred image, or Nothing if processing fails.

EXAMPLE CODE

```
let blurred = ImgProc.BoxAveraging(bitmap, 2);
```

10. ImgProc.BrightContrast(SourceImage, BrightLevel, ContrastLevel, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Adjusts brightness and contrast of image.

SYNTAX

```
let result = ImgProc.BrightContrast(SourceImage, BrightLevel, ContrastLevel, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to adjust.	–
BrightLevel Integer	Brightness adjustment level.	–
ContrastLevel Integer	Contrast adjustment level.	–
IsShowProgress Boolean, Optional	Whether to show progress.	`True`
MaskImage Bitmap, Optional	Optional mask to constrain adjustment.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Bitmap` — Adjusted image, or Nothing if adjustment fails.

EXAMPLE CODE

```
let brightened = ImgProc.BrightContrast(bitmap, 20, 10);
```

11. ImgProc.BW_Dithering_FS(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Applies Floyd-Steinberg dithering for black and white conversion.

SYNTAX

```
let result = ImgProc.BW_Dithering_FS(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to dither.	–
MaskImage Bitmap, Optional	Optional mask to constrain dithering.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Dithered black and white image, or Nothing if processing fails.

EXAMPLE CODE

```
let dithered = ImgProc.BW_Dithering_FS(bitmap);
```

12. ImgProc.BW_Dithering_OrderedDither(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Applies ordered dithering for black and white conversion.

SYNTAX

```
let result = ImgProc.BW_Dithering_OrderedDither(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to dither.	–
MaskImage Bitmap, Optional	Optional mask to constrain dithering.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Dithered black and white image, or Nothing if processing fails.

EXAMPLE CODE

```
let dithered = ImgProc.BW_Dithering_OrderedDither(bitmap);
```

13. ImgProc.CannyEdgeDetecton(SourceImage, LocalBoxSize, SkipGrayScaling, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Detects edges using Canny edge detection algorithm.

SYNTAX

```
let result = ImgProc.CannyEdgeDetecton(SourceImage, LocalBoxSize, SkipGrayScaling, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image for edge detection.	–
LocalBoxSize Integer	Local box size for Canny processing.	–

SkipGrayScaling Boolean, Optional	Whether to skip grayscale conversion.	False
MaskImage Bitmap, Optional	Optional mask to constrain detection.	Nothing
MaskRegionBoundary Rectangle, Optional		Nothing
IsMaskAllRegionSelected Boolean, Optional		True

RETURN VALUE

`Bitmap` — Edge-detected image, or Nothing if detection fails.

EXAMPLE CODE

```
let edges = ImgProc.CannyEdgeDetecton(bitmap, 3);
```

14. ImgProc.ChangeNonBlackToWhite(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Converts all non-black pixels to white, creating binary image.

SYNTAX

```
let result = ImgProc.ChangeNonBlackToWhite(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to process.	–
MaskImage Bitmap, Optional	Optional mask to constrain conversion.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

`Bitmap` — Binary image with white non-black pixels, or Nothing if processing fails.

EXAMPLE CODE

```
let binary = ImgProc.ChangeNonBlackToWhite(bitmap);
```

15. ImgProc.CompositeWithMask(originalImage, processedImage, maskImage, processedRegion)

Returns: Bitmap

DESCRIPTION

Composites processed image with mask on original.

SYNTAX

```
let result = ImgProc.CompositeWithMask(originalImage, processedImage, maskImage, processedRegion);
```

PARAMETERS

Parameter	Description	Default
originalImage Bitmap	Original source image.	–
processedImage Bitmap	Processed result image.	–
maskImage Bitmap	Mask image for blending.	–
processedRegion Rectangle	Region rectangle for processing.	–

RETURN VALUE

`Bitmap` — Composited image, or Nothing if compositing fails.

EXAMPLE CODE

```
let composite = ImgProc.CompositeWithMask(origBitmap, procBitmap, maskBitmap, rect);
```

16. ImgProc.Convert_To_24bitBitmap(SourceImage) Returns: Bitmap

DESCRIPTION

Converts image to 24-bit RGB bitmap.

SYNTAX

```
let result = ImgProc.Convert_To_24bitBitmap(SourceImage);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to convert. Type: Bitmap; required; passed by reference.	–

RETURN VALUE

`Bitmap` — 24-bit RGB bitmap, or Nothing if conversion fails.

EXAMPLE CODE

```
let rgb24 = ImgProc.Convert_To_24bitBitmap(bitmap);
```

17. ImgProc.Convert_To_24bitImage(SourceImage) Returns: Bitmap

DESCRIPTION

Converts to 24-bit RGB image.

SYNTAX

```
let result = ImgProc.Convert_To_24bitImage(SourceImage);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to convert. Type: Bitmap; required; passed by reference.	–

RETURN VALUE

`Bitmap` — 24-bit RGB image, or Nothing if conversion fails.

EXAMPLE CODE

```
let rgb24 = ImgProc.Convert_To_24bitImage(bitmap);
```

18. ImgProc.CorrectUnevenIllumination(SourceImage, GridRows,

GridCols, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Corrects uneven illumination using grid-based division.

SYNTAX

```
let result = ImgProc.CorrectUnevenIllumination(SourceImage, GridRows, GridCols, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to correct.	–
GridRows Integer, Optional	Number of grid rows.	16
GridCols Integer, Optional	Number of grid columns.	16
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Illumination-corrected image, or Nothing if correction fails.

EXAMPLE CODE

```
let corrected = ImgProc.CorrectUnevenIllumination(bitmap, 16, 16);
```

19. ImgProc.Create_BlankImage(SourceImage, BackColor)

Returns: Bitmap

DESCRIPTION

Creates blank image with specified background color.

SYNTAX

```
let result = ImgProc.Create_BlankImage(SourceImage, BackColor);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image defining dimensions.	–
BackColor Color	Background color to fill.	–

RETURN VALUE

Bitmap — Blank image with specified color, or Nothing if creation fails.

EXAMPLE CODE

```
let blank = ImgProc.Create_BlankImage(bitmap, Color.White);
```

20. ImgProc.Create_MaskEllipse(CanvasWidth, CanvasHeighT, MaskRect)

Returns: Bitmap

DESCRIPTION

Creates mask for single ellipse region.

SYNTAX

```
let result = ImgProc.Create_MaskEllipse(CanvasWidth, CanvasHeigh
T, MaskRect);
```

PARAMETERS

Parameter	Description	Default
CanvasWidth Integer	Canvas width.	–
CanvasHeighT Integer	Canvas height.	–
MaskRect Rectangle	Ellipse region for mask.	–

RETURN VALUE

`Bitmap` — Mask bitmap, or Nothing if creation fails.

EXAMPLE CODE

```
let mask = ImgProc.Create_MaskEllipse(800, 600, ellipseRect);
```

21. ImgProc.Create_MaskEllipses(CanvasWidth, CanvasHeighT, MaskRects())

Returns: Bitmap

DESCRIPTION

Creates mask for multiple ellipse regions.

SYNTAX

```
let result = ImgProc.Create_MaskEllipses(CanvasWidth, CanvasHeighT, MaskRects());
```

PARAMETERS

Parameter	Description	Default
`CanvasWidth` Integer	Canvas width.	–
`CanvasHeighT` Integer	Canvas height.	–
`MaskRects()` Rectangle		–

RETURN VALUE

`Bitmap` — Mask bitmap, or Nothing if creation fails.

EXAMPLE CODE

```
let mask = ImgProc.Create_MaskEllipses(800, 600, ellipses());
```

22. ImgProc.Create_MaskRectangle(CanvasWidth, CanvasHeighT, MaskRect)

Returns: Bitmap

DESCRIPTION

Creates mask for single rectangle region.

SYNTAX

```
let result = ImgProc.Create_MaskRectangle(CanvasWidth, CanvasHeigh
T, MaskRect);
```

PARAMETERS

Parameter	Description	Default
`CanvasWidth` Integer	Canvas width.	–
`CanvasHeighT` Integer	Canvas height.	–
`MaskRect` Rectangle	Rectangle region for mask.	–

RETURN VALUE

Bitmap — Mask bitmap, or Nothing if creation fails.

EXAMPLE CODE

```
let mask = ImgProc.Create_MaskRectangle(800, 600, rect);
```

23. ImgProc.Create_MaskRectangles(CanvasWidth, CanvasHeighT, MaskRects())

Returns: Bitmap

DESCRIPTION

Creates mask for multiple rectangle regions.

SYNTAX

```
let result = ImgProc.Create_MaskRectangles(CanvasWidth, CanvasHei
ghT, MaskRects());
```

PARAMETERS

Parameter	Description	Default
`CanvasWidth` Integer	Canvas width.	–
`CanvasHeighT` Integer	Canvas height.	–
`MaskRects()`		–

Rectangle

RETURN VALUE

Bitmap — Mask bitmap, or Nothing if creation fails.

EXAMPLE CODE

```
let mask = ImgProc.Create_MaskRectangles(800, 600, rects());
```

24. ImgProc.Crop(SourceImage, nWidth, nHeight, TargetX, TargetY)

Returns: Bitmap

DESCRIPTION

Crops image to specified dimensions at target position.

SYNTAX

```
let result = ImgProc.Crop(SourceImage, nWidth, nHeight, TargetX, TargetY);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to crop.	–
nWidth Integer	Width of crop area in pixels.	–
nHeight Integer	Height of crop area in pixels.	–
TargetX Integer	X coordinate of top-left corner.	–
TargetY Integer	Y coordinate of top-left corner.	–

RETURN VALUE

Bitmap — Cropped image, or Nothing if cropping fails.

EXAMPLE CODE

```
let cropped = ImgProc.Crop(bitmap, 320, 240, 0, 0);
```

25. ImgProc.DeepCopy(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Creates deep copy of image region.

SYNTAX

```
let result = ImgProc.DeepCopy(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to copy.	–
MaskImage Bitmap, Optional	Optional mask to constrain copy.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Copied image, or Nothing if copy fails.

EXAMPLE CODE

```
let copy = ImgProc.DeepCopy(bitmap);
```

26. ImgProc.Dehaze(SourceImage, PatchRadius, DehazeStrength, SaturationBoost, BrightnessBoost, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Removes atmospheric haze, fog, and chamber glare to restore crisp contrast and vivid colors using a high-speed O(1) Dark Channel Prior algorithm.

SYNTAX

```
let result = ImgProc.Dehaze(SourceImage, PatchRadius, DehazeStrength, SaturationBoost, BrightnessBoost, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to process.	–
PatchRadius Integer, Optional	Local patch radius in pixels for haze estimation (1-150).	25
DehazeStrength Single, Optional	Haze removal intensity (0.0-1.0).	0.7F
SaturationBoost Single, Optional	Color saturation boost after haze removal (-1.0 to 1.0).	0.15F
BrightnessBoost Integer, Optional	Additional brightness compensation (-100 to 100).	10
IsShowProgress Boolean, Optional	Whether to show processing progress.	True
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

`Bitmap` — Processed Bitmap with Dehaze applied, or Nothing if processing fails.

EXAMPLE CODE

```
let clearBmp = ImgProc.Dehaze(bitmap, 25, 0.7F, 0.15F, 10);
```

27. ImgProc.DilateOutlines(SourceGrayImage, BackgroundGray, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

DESCRIPTION

Expands specified outline color on grayscale image edges.

SYNTAX

```
let result = ImgProc.DilateOutlines(SourceGrayImage, BackgroundGray, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceGrayImage Bitmap	Source grayscale image.	–
BackgroundGray Byte	Background grayscale value.	–
MaskImage Bitmap, Optional	Optional mask to constrain processing.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Image with dilated outlines, or Nothing if processing fails.

EXAMPLE CODE

```
let dilated = ImgProc.DilateOutlines(bitmap, 0);
```

28. ImgProc.Draw_Rectangles(Pic_Target, SourceRectangles(), LineColor)

Returns: Bitmap

DESCRIPTION

Draws rectangles on image with specified color.

SYNTAX

```
let result = ImgProc.Draw_Rectangles(Pic_Target, SourceRectangles(), LineColor);
```

PARAMETERS

Parameter	Description	Default

Pic_Target Bitmap	Target image to draw on.	–
SourceRectangles() Rectangle		–
LineColor Color	Color of rectangle outlines.	–

RETURN VALUE

`Bitmap` — Image with drawn rectangles, or Nothing if drawing fails.

EXAMPLE CODE

```
let drawn = ImgProc.Draw_Rectangles(bitmap, rects, Color.Red);
```

29. ImgProc.EdgeDetect_by_ExplicitComp(SourceImage, IsOverlay, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Detects edges using explicit comparison method.

SYNTAX

```
let result = ImgProc.EdgeDetect_by_ExplicitComp(SourceImage, IsOverlay, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image for edge detection.	–
IsOverlay Boolean	Whether to overlay edges on original image.	–
MaskImage Bitmap, Optional	Optional mask to constrain detection.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Bitmap` — Edge-detected image, or Nothing if processing fails.

EXAMPLE CODE

```
let edges = ImgProc.EdgeDetect_by_ExplicitComp(bitmap, true);
```

30. ImgProc.EdgeDetect_by_Homogenity(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Detects edges using homogeneity measure.

SYNTAX

```
let result = ImgProc.EdgeDetect_by_Homogenity(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image for edge detection.	–
MaskImage Bitmap, Optional	Optional mask to constrain detection.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Bitmap` — Edge-detected image, or Nothing if detection fails.

EXAMPLE CODE

```
let edges = ImgProc.EdgeDetect_by_Homogenity(bitmap);
```

31. ImgProc.EdgeDetect_by_LeftHand(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Detects edges using left-hand rule method.

SYNTAX

```
let result = ImgProc.EdgeDetect_by_LeftHand(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image for edge detection.	–
MaskImage Bitmap, Optional	Optional mask to constrain detection.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Edge-detected image, or Nothing if detection fails.

EXAMPLE CODE

```
let edges = ImgProc.EdgeDetect_by_LeftHand(bitmap);
```

32. ImgProc.EdgeDetect_by_Sobel(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Detects edges using Sobel operator.

SYNTAX

```
let result = ImgProc.EdgeDetect_by_Sobel(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default

SourceImage Bitmap	Source image for edge detection.	–
MaskImage Bitmap, Optional	Optional mask to constrain detection.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Edge-detected image, or Nothing if detection fails.

EXAMPLE CODE

```
let edges = ImgProc.EdgeDetect_by_Sobel(bitmap);
```

33. ImgProc.EdgeDetect_by_UpperLeftHand(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Detects edges using upper-left hand kernel.

SYNTAX

```
let result = ImgProc.EdgeDetect_by_UpperLeftHand(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image for edge detection.	–
MaskImage Bitmap, Optional	Optional mask to constrain detection.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

`Bitmap` — Edge-detected image, or Nothing if detection fails.

EXAMPLE CODE

```
let edges = ImgProc.EdgeDetect_by_UpperLeftHand(bitmap);
```

34. ImgProc.Emboss(SourceImage, Angle, Distance, Strength, Bias, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Applies emboss filter to create 3D relief effect.

SYNTAX

```
let result = ImgProc.Emboss(SourceImage, Angle, Distance, Strength, Bias, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to emboss.	–
Angle Single, Optional	Direction of light source in degrees (0-360).	135.0F
Distance Integer, Optional	Distance of emboss effect.	1
Strength Single, Optional	Strength of emboss effect.	1.0F
Bias Integer, Optional	Bias value for emboss calculation.	128
MaskImage Bitmap, Optional	Optional mask to constrain effect.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

`Bitmap` — Embossed image, or Nothing if processing fails.

EXAMPLE CODE

```
let embossed = ImgProc.Emboss(bitmap, 135.0, 1, 1.0, 128);
```

35. ImgProc.ErodeOutlines(SourceGrayImage, BackgroundGray, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Removes specified outline color from grayscale image edges.

SYNTAX

```
let result = ImgProc.ErodeOutlines(SourceGrayImage, BackgroundGray, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceGrayImage Bitmap	Source grayscale image.	–
BackgroundGray Byte	Background grayscale value.	–
MaskImage Bitmap, Optional	Optional mask to constrain processing.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Bitmap` — Image with eroded outlines, or Nothing if processing fails.

EXAMPLE CODE

```
let eroded = ImgProc.ErodeOutlines(bitmap, 0);
```

36. ImgProc.Flip_Horizontal(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Flips image horizontally (mirror left-right), either entirely or within a specified mask region.

SYNTAX

```
let result = ImgProc.Flip_Horizontal(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to flip.	–
MaskImage Bitmap, Optional	Mask image defining the active region (blue channel = 0 for active pixels).	Nothing
MaskRegionBoundary Rectangle, Optional	Bounding rectangle of the selected mask region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	True to flip the entire image; False to flip only within the masked region.	True

RETURN VALUE

Bitmap — Horizontally flipped image, or Nothing if flipping fails.

EXAMPLE CODE

```
let flipped = ImgProc.Flip_Horizontal(bitmap);
```

37. ImgProc.Flip_Vertical(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Flips image vertically (mirror top-bottom), either entirely or within a specified mask region.

SYNTAX

```
let result = ImgProc.Flip_Vertical(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to flip.	–
MaskImage Bitmap, Optional	Mask image defining the active region (blue channel = 0 for active pixels).	Nothing
MaskRegionBoundary Rectangle, Optional	Bounding rectangle of the selected mask region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	True to flip the entire image; False to flip only within the masked region.	True

RETURN VALUE

Bitmap — Vertically flipped image, or Nothing if flipping fails.

EXAMPLE CODE

```
let flipped = ImgProc.Flip_Vertical(bitmap);
```

38. ImgProc.GammaAdjustment(SourceImage, gammaValue, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Applies gamma adjustment to modify the overall brightness and contrast of the image.

SYNTAX

```
let result = ImgProc.GammaAdjustment(SourceImage, gammaValue, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage	Source image for gamma	–

Bitmap	adjustment.	
gammaValue Double	Gamma value for adjustment.	–
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Gamma-adjusted image, or Nothing if adjustment fails.

EXAMPLE CODE

```
let adjusted = ImgProc.GammaAdjustment(bitmap, 1.5);
```

39. ImgProc.GaussianFilter(SourceImage, FilterSize, WeightFactor, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Applies Gaussian blur filter with configurable size.

SYNTAX

```
let result = ImgProc.GaussianFilter(SourceImage, FilterSize, WeightFactor, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to blur.	–
FilterSize Integer	Size of Gaussian kernel.	–
WeightFactor Integer, Optional	Weight factor for Gaussian (0 for default).	0

MaskImage Bitmap, Optional	Optional mask to constrain blur.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Blurred image, or Nothing if processing fails.

EXAMPLE CODE

```
let blurred = ImgProc.GaussianFilter(bitmap, 3, 0);
```

40. ImgProc.Generate_Canvas(nWidth, nHeight, BackColor)

Returns: Bitmap

DESCRIPTION

Creates blank canvas with specified dimensions and background color.

SYNTAX

```
let result = ImgProc.Generate_Canvas(nWidth, nHeight, BackColor);
```

PARAMETERS

Parameter	Description	Default
nWidth Integer	Canvas width in pixels.	–
nHeight Integer	Canvas height in pixels.	–
BackColor Color	Background color for canvas.	–

RETURN VALUE

Bitmap — Blank canvas image, or Nothing if creation fails.

EXAMPLE CODE

```
let canvas = ImgProc.Generate_Canvas(640, 480, Color.White);
```

41. ImgProc.HDRLocalToneMapping(SourceImage, BlurRadius, DetailStrength, ToneCompression, BrightnessBoost, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Enhances local contrast and compresses dynamic range (shadow/highlight recovery) using a high-speed O(1) Stack Blur algorithm.

SYNTAX

```
let result = ImgProc.HDRLocalToneMapping(SourceImage, BlurRadius, DetailStrength, ToneCompression, BrightnessBoost, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to process.	–
BlurRadius Integer, Optional	Local neighborhood radius in pixels (1-150).	40
DetailStrength Single, Optional	Clarity and fine-detail amplification strength (0.0-3.0).	1.2F
ToneCompression Single, Optional	Shadow/highlight dynamic range compression strength (0.0-1.0).	0.5F
BrightnessBoost Integer, Optional	Additional brightness boost applied to the final pixels (-100 to 100).	0
IsShowProgress Boolean, Optional	Whether to show processing progress.	True
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected	Whether entire region is	True

Boolean, Optional	selected.

RETURN VALUE

`Bitmap` — Processed Bitmap with HDR Local Tone Mapping applied, or Nothing if processing fails.

EXAMPLE CODE

```
let hdrBmp = ImgProc.HDRLocalToneMapping(bitmap, 40, 1.2F, 0.5F, 0);
```

42. ImgProc.HighlightBloomHalation(SourceImage, SpreadRadius, Threshold, BloomStrength, HalationTint, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Applies selective optical highlight bloom and analog cinema film halation to bright light sources using a high-speed O(1) Stack Blur engine.

SYNTAX

```
let result = ImgProc.HighlightBloomHalation(SourceImage, SpreadRadius, Threshold, BloomStrength, HalationTint, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to process.	–
SpreadRadius Integer, Optional	Diffusion radius in pixels (1-150).	25
Threshold Integer, Optional	Luminance threshold for isolating bright highlights (0-250).	180
BloomStrength Single, Optional	Intensity of the highlight bloom glow (0.0-2.0).	0.8F
HalationTint Single, Optional	Red-orange cinema film halation tint ratio (0.0-1.0).	0.0F
IsShowProgress Boolean, Optional	Whether to show processing progress	True

MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Processed Bitmap with Highlight Bloom & Halation applied, or Nothing if processing fails.

EXAMPLE CODE

```
let bloomBmp = ImgProc.HighlightBloomHalation(bitmap, 25, 180, 0.8F, 0.35F);
```

43. ImgProc.HighPassSharpening(SourceImage, Radius, Strength, Threshold, BlendMode, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Sharpens multi-scale structural edges and surface textures using a high-speed O(1) High-Pass frequency extraction algorithm.

SYNTAX

```
let result = ImgProc.HighPassSharpening(SourceImage, Radius, Strength, Threshold, BlendMode, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to process.	–
Radius Integer, Optional	Frequency radius in pixels (1-100).	10
Strength Single, Optional	High-frequency detail amplification strength (0.0-2.0).	1.0F

Threshold Integer, Optional	Noise suppression threshold in pixel levels (0-50).	3
BlendMode Integer, Optional	Integer representing the blending algorithm. Supported values: 0: Overlay (Balanced) 1: SoftLight (Subtle) 2: HardLight (Intense)	0
IsShowProgress Boolean, Optional	Whether to show processing progress.	True
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

`Bitmap` — Processed Bitmap with High-Pass Sharpening applied, or Nothing if processing fails.

EXAMPLE CODE

```
let sharpBmp = ImgProc.HighPassSharpening(bitmap, 10, 1.0F, 3, 0);
```

44. ImgProc.HoughTransform_For_Circle(SourceImage, Radius, AngleCount, IsAbsoluteNormalization, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Detects circles in image using Hough transform.

SYNTAX

```
let result = ImgProc.HoughTransform_For_Circle(SourceImage, Radius, AngleCount, IsAbsoluteNormalization, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image for circle detection. Type: Bitmap; required; passed by reference.	–
Radius Single	Circle radius to detect.	–
AngleCount Integer	Number of angles to check.	–
IsAbsoluteNormalization Boolean	Whether to use absolute normalization.	–
MaskImage Bitmap, Optional	Optional mask to constrain detection.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Bitmap` — Image with detected circles, or Nothing if detection fails.

EXAMPLE CODE

```
let circles = ImgProc.HoughTransform_For_Circle(bitmap, 50.0, 360, true);
```

45. ImgProc.HoughTransform_For_Line(SourceImage, UserMaxCount, IsAbsoluteNormalization, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Detects lines in image using Hough transform.

SYNTAX

```
let result = ImgProc.HoughTransform_For_Line(SourceImage, UserMaxCount, IsAbsoluteNormalization, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image for line detection. Type: Bitmap; required; passed by reference.	–
UserMaxCount Integer	Maximum line count to detect.	–
IsAbsoluteNormalization Boolean	Whether to use absolute normalization.	–
MaskImage Bitmap, Optional	Optional mask to constrain detection.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Image with detected lines, or Nothing if detection fails.

EXAMPLE CODE

```
let lines = ImgProc.HoughTransform_For_Line(bitmap, 10, true);
```

46. ImgProc.Image_FromFile(FileName) Returns: Bitmap

DESCRIPTION

Loads image from file path.

SYNTAX

```
let result = ImgProc.Image_FromFile(FileName);
```

PARAMETERS

Parameter	Description	Default
FileName String	Path to image file.	–

RETURN VALUE

`Bitmap` — Loaded image as Bitmap, or Nothing if loading fails.

EXAMPLE CODE

```
let bitmap = ImgProc.Image_FromFile("C:\Images\photo.jpg");
```

47. ImgProc.ImageStamp(SourceImage, StampImage, destRectX, destRectY, destRectWidth, destRectHeight, angle, opacityPercent, enableTransparentColor, transparentColorArgb, enableFeather, featherRadius, enableGradualAlpha, tolerance)

Returns: Bitmap

DESCRIPTION

Stamps overlay image on source with transformation.

SYNTAX

```
let result = ImgProc.ImageStamp(SourceImage, StampImage, destRectX, destRectY, destRectWidth, destRectHeight, angle, opacityPercent, enableTransparentColor, transparentColorArgb, enableFeather, featherRadius, enableGradualAlpha, tolerance);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to stamp on.	–
StampImage Bitmap	Stamp image to overlay.	–
destRectX Single	Destination rectangle X.	–
destRectY Single	Destination rectangle Y.	–
destRectWidth Single	Destination rectangle width.	–
destRectHeight Single	Destination rectangle height.	–
angle Single, Optional	Rotation angle in degrees.	0.0F
opacityPercent	Opacity percentage (0-100)	100

opacityPercent Integer, Optional	Opacity percentage (0-100).	100
enableTransparentColor Boolean, Optional	Whether to enable transparent color.	False
transparentColorArgb Integer, Optional	ARGB color value for transparency.	-1
enableFeather Boolean, Optional	Whether to enable edge feathering.	False
featherRadius Integer, Optional	Feather radius in pixels.	10
enableGradualAlpha Boolean, Optional	Whether to enable gradual alpha.	False
tolerance Integer, Optional	Color tolerance threshold.	30

RETURN VALUE

`Bitmap` — Stamped image, or Nothing if stamping fails.

EXAMPLE CODE

```
let stamped = ImgProc.ImageStamp(bitmap, stampBitmap, 10, 10, 100, 100, 0, 100);
```

48. ImgProc.Invert(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Inverts image colors (photographic negative).

SYNTAX

```
let result = ImgProc.Invert(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to invert.	–

MaskImage Bitmap, Optional	Optional mask to constrain inversion.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Inverted image, or Nothing if inversion fails.

EXAMPLE CODE

```
let inverted = ImgProc.Invert(bitmap);
```

49. ImgProc.ManualBinarization(SourceImage, ThresholdValue, SkipGrayScaling, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Converts image to binary using manual threshold value.

SYNTAX

```
let result = ImgProc.ManualBinarization(SourceImage, ThresholdValue, SkipGrayScaling, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to binarize.	–
ThresholdValue Integer	Threshold value for binarization (0-255).	–
SkipGrayScaling Boolean, Optional	Whether to skip grayscale conversion.	False
IsShowProgress Boolean, Optional	Whether to display progress.	True
MaskImage	Optional mask to constrain	Nothing

Bitmap, Optional	binarization.	
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Binary image, or Nothing if binarization fails.

EXAMPLE CODE

```
let binary = ImgProc.ManualBinarization(bitmap, 128);
```

50. ImgProc.MiniatureEffect(SourceImage, BlurRadius, FocusPosition, FocusSize, SaturationBoost, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Simulates a tilt-shift miniature lens effect with smooth depth-of-field blurring and saturation enhancement.

SYNTAX

```
let result = ImgProc.MiniatureEffect(SourceImage, BlurRadius, FocusPosition, FocusSize, SaturationBoost, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to process.	–
BlurRadius Integer, Optional	Out-of-focus blur radius in pixels (1-150).	25
FocusPosition Single, Optional	Vertical center position of the sharp focus zone (0.0-1.0).	0.5F
FocusSize Single, Optional	Relative height of the sharp focus band (0.05-0.80).	0.25F
SaturationBoost	Color vibrancy boost (-1.0 to	0.25F

Single, Optional	1.0).	
IsShowProgress Boolean, Optional	Whether to show processing progress.	True
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Processed Bitmap with Miniature Effect applied, or Nothing if processing fails.

EXAMPLE CODE

```
let miniBmp = ImgProc.MiniatureEffect(bitmap, 25, 0.5F, 0.25F, 0.25F);
```

51. ImgProc.MosaicFilter(SourceImage, isPixelMode, value, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Applies mosaic or pixelation effect to blur regions.

SYNTAX

```
let result = ImgProc.MosaicFilter(SourceImage, isPixelMode, value, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to process.	–
isPixelMode Boolean	Whether to use pixel mode.	–
value Integer	Mosaic block size or pixel value.	–

MaskImage Bitmap, Optional	Optional mask to constrain effect.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Mosaiced image, or Nothing if processing fails.

EXAMPLE CODE

```
let mosaiced = ImgProc.MosaicFilter(bitmap, true, 10);
```

52. ImgProc.MultibandColor(original, rSat, rInt, ySat, yInt, gSat, gInt, cSat, cyanInt, bSat, bInt, mSat, mInt, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Applies multiband color processing adjusting saturation and intensity by color band.

SYNTAX

```
let result = ImgProc.MultibandColor(original, rSat, rInt, ySat, yInt, gSat, gInt, cSat, cyanInt, bSat, bInt, mSat, mInt, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
original Bitmap	Source image to process.	–
rSat Integer	Red saturation adjustment (-100 to 100).	–
rInt Integer	Red intensity adjustment (-100 to 100).	–
ySat Integer	Yellow saturation adjustment (-100 to 100).	–

yInt Integer	Yellow intensity adjustment (-100 to 100).	–
gSat Integer	Green saturation adjustment (-100 to 100).	–
gInt Integer	Green intensity adjustment (-100 to 100).	–
cSat Integer	Cyan saturation adjustment (-100 to 100).	–
cyanInt Integer	Cyan intensity adjustment (-100 to 100).	–
bSat Integer	Blue saturation adjustment (-100 to 100).	–
bInt Integer	Blue intensity adjustment (-100 to 100).	–
mSat Integer	Magenta saturation adjustment (-100 to 100).	–
mInt Integer	Magenta intensity adjustment (-100 to 100).	–
MaskImage Image, Optional	Optional mask to constrain processing.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Bitmap` — Color-adjusted image, or Nothing if processing fails.

EXAMPLE CODE

```
let mbcolor = ImgProc.MultibandColor(bitmap, 10, 5, -5, 10, 0, 0, 5, 0, -10, 5, 0, 0);
```

53. ImgProc.OrtonEffect(SourceImage, BlurRadius, GlowStrength, ExposureBoost, BrightnessBoost, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Applies a dreamy, painterly luminescence (Orton Effect) by combining overexposed sharp and blurred layers via Multiply blending.

SYNTAX

```
let result = ImgProc.OrtonEffect(SourceImage, BlurRadius, GlowStrength, ExposureBoost, BrightnessBoost, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to process.	–
BlurRadius Integer, Optional	Blur radius for the glow diffusion in pixels (1-150).	25
GlowStrength Single, Optional	Overall blending opacity of the Orton effect (0.0-1.0).	0.6F
ExposureBoost Single, Optional	Highlight overexposure boost in the Screen pass (0.0-1.0).	0.7F
BrightnessBoost Integer, Optional	Additional brightness boost applied to the final pixels (-100 to 100).	0
IsShowProgress Boolean, Optional	Whether to show processing progress.	True
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

`Bitmap` — Processed Bitmap with Orton Effect applied, or Nothing if processing fails.

EXAMPLE CODE

```
let ortonBmp = ImgProc.OrtonEffect(bitmap, 25, 0.6F, 0.7F, 0);
```

54. ImgProc.Outline(SourceImage, InvertBinarization, ApplyBinarization, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Applies edge detection (outline) filter to extract object boundaries while always preserving the original pixel colors on the extracted outlines.

SYNTAX

```
let result = ImgProc.Outline(SourceImage, InvertBinarization, ApplyBinarization, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image for edge detection.	–
InvertBinarization Boolean, Optional	Whether to invert the binarization or background result (e.g., dark objects on a light background).	False
ApplyBinarization Boolean, Optional	Whether to apply threshold-based pre-binarization for boundary detection. Set to False to skip binarization and extract boundaries directly from all non-zero color regions (e.g., after Region Coloring) without brightness threshold loss. Original outline colors are preserved in both modes.	True
IsShowProgress Boolean, Optional	Whether to display progress.	False
MaskImage Bitmap, Optional	Optional mask to constrain edge detection.	Nothing
MaskRegionBoundary	Boundary rectangle for	Nothing

Rectangle, Optional	masked region.	
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Edge-detected image with original pixel colors preserved on the outlines, or Nothing if processing fails.

EXAMPLE CODE

```
let edges = ImgProc.Outline(bitmap, false, false, false);
```

55. ImgProc.Overlay_Mask3DByte_To_Image(SourceImage, SourceMaskImage3DByte, TransparentColor, Opacity, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Overlays 3D byte mask to image with transparency.

SYNTAX

```
let result = ImgProc.Overlay_Mask3DByte_To_Image(SourceImage, SourceMaskImage3DByte, TransparentColor, Opacity, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to overlay. Type: Bitmap; required; passed by reference.	–
SourceMaskImage3DByte Byte(,,)	3D byte array mask [height, width, channels]. Type: Byte(,,); required; passed by reference.	–
TransparentColor Integer	Color to treat as transparent.	–
Opacity Single	Opacity level (0.0-1.0).	–
MaskImage	Optional mask to constrain	Nothing

Bitmap, Optional	operation.	
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Overlaid image, or Nothing if overlay fails.

EXAMPLE CODE

```
let overlaid = ImgProc.Overlay_Mask3DByte_To_Image(bitmap,
maskArray, RGB(255,255,255), 0.5);
```

56. ImgProc.Overlay_MaskImage2DArray_To_Image3DArray(SourceImage2DArray(, MaskImage2DArray, MaskRed, MaskGreen, MaskBlue, x1, y1, TransparentColor)

Returns: Void

DESCRIPTION

Overlays 2D mask array on 3D image array.

SYNTAX

```
ImgProc.Overlay_MaskImage2DArray_To_Image3DArray(SourceImage2DArray(,,), MaskImage2DArray, MaskRed, MaskGreen, MaskBlue, x1, y1, TransparentColor);
```

PARAMETERS

Parameter	Description	Default
SourceImage2DArray(,,) Byte		–
MaskImage2DArray Byte(,)	2D byte array mask [height, width]. Type: Byte(,); required; passed by reference.	–
MaskRed Byte	Red channel value for mask.	–
MaskGreen	Green channel value for mask.	–

Byte		
MaskBlue Byte	Blue channel value for mask.	–
x1 Integer	Start X coordinate.	–
y1 Integer	Start Y coordinate.	–
TransparentColor Byte	Transparent color value.	–

RETURN VALUE

`Void` — None.

EXAMPLE CODE

```
ImgProc.Overlay_MaskImage2DArray_To_Image3DArray(imageArray,
maskArray, 255, 0, 0, 0, 0, 255);
```

57. ImgProc.Overlay_MaskImage_To_Image(SourceImage, SourceMaskImage, TransparentColor, Opacity)

Returns: Bitmap

DESCRIPTION

Overlays mask bitmap to image with transparency.

SYNTAX

```
let result = ImgProc.Overlay_MaskImage_To_Image(SourceImage, Sour
ceMaskImage, TransparentColor, Opacity);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to overlay. Type: Bitmap; required; passed by reference.	–
SourceMaskImage Bitmap	Mask image to overlay. Type: Bitmap; required; passed by reference.	–
TransparentColor Integer	Color to treat as transparent.	–

Opacity Single	Opacity level (0.0-1.0).	–

RETURN VALUE

`Bitmap` — Overlaid image, or Nothing if overlay fails.

EXAMPLE CODE

```
let overlaid = ImgProc.Overlay_MaskImage_To_Image(bitmap,
maskBitmap, RGB(255,255,255), 0.5);
```

58. ImgProc.OverlayImage(SourceImage, overlayImg, transR, transG, transB, opacity, isFeather, featherRadius, tolerance, isGradual, MaskImage, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Overlays another image with transparency and feathering.

SYNTAX

```
let result = ImgProc.OverlayImage(SourceImage, overlayImg, trans
R, transG, transB, opacity, isFeather, featherRadius, tolerance,
isGradual, MaskImage, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image for overlay.	–
overlayImg Bitmap	Image to overlay on source.	–
transR Integer, Optional	Transparent red component (0-255).	255
transG Integer, Optional	Transparent green component (0-255).	255
transB Integer, Optional	Transparent blue component (0-255).	255
opacity Single, Optional	Opacity factor (0.0-1.0).	1.0F

isFeather Boolean, Optional	Whether to apply edge feathering.	False
featherRadius Integer, Optional	Feather radius in pixels.	10
tolerance Integer, Optional	Color tolerance threshold (0-255).	0
isGradual Boolean, Optional	Whether to apply gradual blending.	False
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Overlaid image, or Nothing if operation fails.

EXAMPLE CODE

```
let overlaid = ImgProc.OverlayImage(bitmap, overlayBitmap, 255, 255, 255, 1.0, false, 10);
```

59. ImgProc.OverlayMask_Parallel(sourceImage, maskImage, overlayColor, alpha)

Returns: Bitmap

DESCRIPTION

Overlays mask with color and alpha transparency.

SYNTAX

```
let result = ImgProc.OverlayMask_Parallel(sourceImage, maskImage, overlayColor, alpha);
```

PARAMETERS

Parameter	Description	Default
sourceImage Bitmap	Source image to overlay.	–
maskImage	Mask image.	–

Bitmap		
overlayColor Color	Color for overlay.	–
alpha Integer	Alpha transparency value.	–

RETURN VALUE

`Bitmap` — Overlaid image, or Nothing if overlay fails.

EXAMPLE CODE

```
let overlaid = ImgProc.OverlayMask_Parallel(bitmap, maskBitmap,
Color.Red, 128);
```

60. ImgProc.PasteAndFitToRectangle(BaseImage, OverlayImage, TargetRect)

Returns: Bitmap

DESCRIPTION

Pastes and fits overlay image into target rectangle.

SYNTAX

```
let result = ImgProc.PasteAndFitToRectangle(BaseImage, OverlayIma
ge, TargetRect);
```

PARAMETERS

Parameter	Description	Default
BaseImage Bitmap	Base image to paste on.	–
OverlayImage Bitmap	Overlay image to paste.	–
TargetRect Rectangle	Target rectangle for pasting.	–

RETURN VALUE

`Bitmap` — Pasted and fitted image, or Nothing if operation fails.

EXAMPLE CODE

```
let pasted = ImgProc.PasteAndFitToRectangle(baseImage,
overlayImage, New Rectangle(10, 10, 100, 100));
```

61. ImgProc.Remap(SourceImage, mapX, mapY, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Remaps image using coordinate maps.

SYNTAX

```
let result = ImgProc.Remap(SourceImage, mapX, mapY, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to remap.	–
mapX Single(,)	X coordinate map.	–
mapY Single(,)	Y coordinate map.	–
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Remapped image, or Nothing if remapping fails.

EXAMPLE CODE

```
let remapped = ImgProc.Remap(bitmap, xMap, yMap);
```

62. ImgProc.RemoveBackground(SourceImage, MatteImage, MaskRegionBoundary, IsMaskAllRegionSelected, BgColor, NewBackgroundImage)

Returns: Bitmap

DESCRIPTION

Removes background using matte with optional replacement.

SYNTAX

```
let result = ImgProc.RemoveBackground(SourceImage, MatteImage, MaskRegionBoundary, IsMaskAllRegionSelected, BgColor, NewBackgroundImage);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to process.	–
MatteImage Bitmap	Matte/alpha mask image.	–
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True
BgColor Color, Optional	Background color replacement.	Nothing
NewBackgroundImage Bitmap, Optional	New background image to composite.	Nothing

RETURN VALUE

Bitmap — Image with background removed, or Nothing if operation fails.

EXAMPLE CODE

```
let noBackground = ImgProc.RemoveBackground(bitmap, matteBitmap);
```

63. ImgProc.Resize(SourceImage, nWidth, nHeight, InterpolationMode)

Returns: Bitmap

DESCRIPTION

Resizes image to specified width and height.

SYNTAX

```
let result = ImgProc.Resize(SourceImage, nWidth, nHeight, Interpo
lationMode);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to resize.	–
nWidth Integer	Target width in pixels.	–
nHeight Integer	Target height in pixels.	–
InterpolationMode Drawing2D.InterpolationMode, Optional	Enum representing the interpolation method. Type: Drawing2D.InterpolationMode; optional; default: HighQualityBicubic. Supported InterpolationMode values: 0: Default 1: Low 2: High 3: Bilinear 4: Bicubic 5: NearestNeighbor 6: HighQualityBilinear 7: HighQualityBicubic	Drawing2D.Interpo

RETURN VALUE

Bitmap — Resized image, or Nothing if resizing fails.

EXAMPLE CODE

```
let resized = ImgProc.Resize(bitmap, 640, 480);
```

64. ImgProc.Resize_Canvas(SourceImage, nWidth, nHeight, StretchDirection, BackColor)

Returns: Bitmap

DESCRIPTION

Resizes canvas and places the image within new dimensions.

SYNTAX

```
let result = ImgProc.Resize_Canvas(SourceImage, nWidth, nHeight, StretchDirection, BackColor);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to resize.	–
nWidth Integer	New canvas width.	–
nHeight Integer	New canvas height.	–
StretchDirection String	Image placement on the resized canvas. Supported values: "LeftTop": Place image at the top-left corner "RightTop": Place image at the top-right corner "LeftBottom": Place image at the bottom-left corner "RightBottom": Place image at the bottom-right corner Any other value (e.g., "Center"): Place image at the center	–
BackColor Color	Background fill color.	–

RETURN VALUE

`Bitmap` — Resized canvas image, or Nothing if resizing fails.

EXAMPLE CODE

```
let resized = ImgProc.Resize_Canvas(bitmap, 640, 480, "Center", Color.White);
```

65. ImgProc.ResizeByZoomFactor(SourceImage, ZoomFactor, InterpolationMode)

Returns: Bitmap

DESCRIPTION

Resizes image by specified zoom factor (e.g., 0.5 for 50%, 2.0 for 200%).

SYNTAX

```
let result = ImgProc.ResizeByZoomFactor(SourceImage, ZoomFactor, InterpolationMode);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to resize.	–
ZoomFactor Single	Zoom factor multiplier.	–
InterpolationMode Drawing2D.InterpolationMode, Optional	Enum representing the interpolation method. Type: Drawing2D.InterpolationMode; optional; default: HighQualityBicubic. Supported InterpolationMode values: 0: Default 1: Low 2: High 3: Bilinear 4: Bicubic 5: NearestNeighbor 6: HighQualityBilinear 7: HighQualityBicubic	Drawing2D.Interpo

RETURN VALUE

Bitmap — Resized image, or Nothing if resizing fails.

EXAMPLE CODE

```
let resized = ImgProc.ResizeByZoomFactor(bitmap, 0.5);
```

66. ImgProc.RollingBoxBackgroundSubtraction(SourceImage, Radius, IsAutomatic, IsObjectWhite, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Removes background using rolling box filter.

SYNTAX

```
let result = ImgProc.RollingBoxBackgroundSubtraction(SourceImage, Radius, IsAutomatic, IsObjectWhite, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to process.	–
Radius Integer	Rolling box radius.	–
IsAutomatic Boolean, Optional	Whether to use automatic mode.	False
IsObjectWhite Boolean, Optional	Whether object is white or black.	True
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Background-subtracted image, or Nothing if operation fails.

EXAMPLE CODE

```
let subtracted = ImgProc.RollingBoxBackgroundSubtraction(bitmap, 50);
```

67. ImgProc.Rotate(SourceImage, CenterX, CenterY, AngleDegree, BackgroundColor, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Rotates image around specified center point by angle in degrees, either entirely or within a specified mask region.

SYNTAX

```
let result = ImgProc.Rotate(SourceImage, CenterX, CenterY, AngleDegree, BackgroundColor, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to rotate.	–
CenterX Integer	X coordinate of rotation center.	–
CenterY Integer	Y coordinate of rotation center.	–
AngleDegree Single	Rotation angle in degrees.	–
BackgroundColor Color	Fill color for empty areas after rotation.	–
MaskImage Bitmap, Optional	Mask image defining the active region (blue channel = 0 for active pixels).	`Nothing`
MaskRegionBoundary Rectangle, Optional	Bounding rectangle of the selected mask region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	True to rotate the entire image; False to rotate only within the masked region.	`True`

RETURN VALUE

`Bitmap` — Rotated image, or Nothing if rotation fails.

EXAMPLE CODE

```
let rotated = ImgProc.Rotate(bitmap, 320, 240, 45.0, Color.White);
```

68. ImgProc.RotateAtCenter(SourceImage, AngleDegree)

Returns: Bitmap

DESCRIPTION

Rotates image around its center by specified angle in degrees.

SYNTAX

```
let result = ImgProc.RotateAtCenter(SourceImage, AngleDegree);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to rotate.	–
AngleDegree Single	Rotation angle in degrees.	–

RETURN VALUE

Bitmap — Rotated image, or Nothing if rotation fails.

EXAMPLE CODE

```
let rotated = ImgProc.RotateAtCenter(bitmap, 45.0);
```

69. ImgProc.RotateFlip(SourceImage, RotateFlipOption)

Returns: Bitmap

DESCRIPTION

Applies rotation and flip transformation to image.

SYNTAX

```
let result = ImgProc.RotateFlip(SourceImage, RotateFlipOption);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to transform.	–
RotateFlipOption RotateFlipType	Type of rotation and flip operation. Supported values:	–

0: RotateNoneFlipNone - No rotation, no flip
1: Rotate90FlipNone - Rotate 90 degrees clockwise
2: Rotate180FlipNone - Rotate 180 degrees
3: Rotate270FlipNone - Rotate 270 degrees clockwise
4: RotateNoneFlipX - Flip horizontally
5: Rotate90FlipX - Rotate 90 degrees, then flip horizontally
6: Rotate180FlipX - Rotate 180 degrees, then flip horizontally (same as flip vertically)
7: Rotate270FlipX - Rotate 270 degrees, then flip horizontally

RETURN VALUE

`Bitmap` — Transformed image, or Nothing if transformation fails.

EXAMPLE CODE

```
let transformed = ImgProc.RotateFlip(bitmap, RotateFlipType.Rotate90FlipNone);
```

70. ImgProc.ShadowsHighlights(SourceImage, ShadowAmount, HighlightAmount, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Adjusts shadows and highlights separately.

SYNTAX

```
let result = ImgProc.ShadowsHighlights(SourceImage, ShadowAmount, HighlightAmount, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to adjust.	–
ShadowAmount	Shadow adjustment amount.	50.0F

Single, Optional		
HighlightAmount Single, Optional	Highlight adjustment amount.	0.0F
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Adjusted image, or Nothing if adjustment fails.

EXAMPLE CODE

```
let adjusted = ImgProc.ShadowsHighlights(bitmap, 50.0, 0.0);
```

71. ImgProc.Sharpen_3x3(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Sharpens image using basic 3x3 kernel.

SYNTAX

```
let result = ImgProc.Sharpen_3x3(SourceImage, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to sharpen.	–
MaskImage Bitmap, Optional	Optional mask to constrain sharpening.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected	Whether entire region is	True

	Boolean, Optional	selected.

RETURN VALUE

`Bitmap` — Sharpened image, or Nothing if processing fails.

EXAMPLE CODE

```
let sharpened = ImgProc.Sharpen_3x3(bitmap);
```

72. ImgProc.Sharpen_3x3_Improved(SourceImage, Strength, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Sharpens image using improved 3x3 kernel with adjustable strength.

SYNTAX

```
let result = ImgProc.Sharpen_3x3_Improved(SourceImage, Strength, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to sharpen.	–
Strength Single, Optional	Sharpening strength factor.	`1.0F`
MaskImage Bitmap, Optional	Optional mask to constrain sharpening.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Bitmap` — Sharpened image, or Nothing if processing fails.

EXAMPLE CODE

```
let sharpened = ImgProc.Sharpen_3x3_Improved(bitmap, 1.5);
```

73. ImgProc.Sharpen_5x5_HighQuality_Fixed(SourceImage, Strength, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Sharpens image using high-quality 5x5 kernel.

SYNTAX

```
let result = ImgProc.Sharpen_5x5_HighQuality_Fixed(SourceImage, Strength, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to sharpen.	–
Strength Single, Optional	Sharpening strength factor.	1.0F
MaskImage Bitmap, Optional	Optional mask to constrain sharpening.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — High-quality sharpened image, or Nothing if processing fails.

EXAMPLE CODE

```
let sharpened = ImgProc.Sharpen_5x5_HighQuality_Fixed(bitmap, 1.2);
```

74. ImgProc.Sharpen_UnsharpMask(SourceImage, FilterSize, AmountPercent, Threshold, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Sharpens image using unsharp mask technique.

SYNTAX

```
let result = ImgProc.Sharpen_UnsharpMask(SourceImage, FilterSize, AmountPercent, Threshold, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to sharpen.	–
FilterSize Integer, Optional	Filter kernel size.	11
AmountPercent Integer, Optional	Sharpening amount percentage.	100
Threshold Integer, Optional	Edge detection threshold.	0
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

`Bitmap` — Sharpened image, or Nothing if sharpening fails.

EXAMPLE CODE

```
let sharpened = ImgProc.Sharpen_UnsharpMask(bitmap, 11, 100, 0);
```

75. ImgProc.SmartDenoise(SourceImage, Radius, Threshold, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Applies smart noise reduction preserving edges.

SYNTAX

```
let result = ImgProc.SmartDenoise(SourceImage, Radius, Threshold, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to denoise.	–
Radius Integer, Optional	Filter radius.	5
Threshold Integer, Optional	Edge preservation threshold.	30
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Denoised image, or Nothing if denoising fails.

EXAMPLE CODE

```
let denoised = ImgProc.SmartDenoise(bitmap, 5, 30);
```

76. ImgProc.SmartSketch(SourceImage, StrokeWidth, PencilDarkness, ColorBlend, PaperBrightness, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Converts the image into a fine pencil sketch or colored-pencil technical line art using a high-speed O(1) Color Dodge algorithm.

SYNTAX

```
let result = ImgProc.SmartSketch(SourceImage, StrokeWidth, PencilDarkness, ColorBlend, PaperBrightness, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to process.	–
StrokeWidth Integer, Optional	Stroke width (blur radius) in pixels (1-100).	15
PencilDarkness Single, Optional	Graphite line darkness and shading intensity (0.01-1.0).	0.5F
ColorBlend Single, Optional	Original color blending ratio (0.0 for pure grayscale sketch, up to 1.0 for colored-pencil look).	0.0F
PaperBrightness Integer, Optional	Background paper brightness adjustment (-100 to 100).	0
IsShowProgress Boolean, Optional	Whether to show processing progress.	True
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

`Bitmap` — Processed Bitmap with Smart Sketch applied, or Nothing if processing fails.

EXAMPLE CODE

```
let sketchBmp = ImgProc.SmartSketch(bitmap, 15, 0.5F, 0.0F, 0);
```

77. ImgProc.SoftGlow(SourceImage, BlurRadius, GlowStrength, BrightnessBoost, BlendMode, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Applies a dreamy, radiant luminescence effect using a high-speed O(1) Stack Blur and Screen/SoftLight blending.

SYNTAX

```
let result = ImgProc.SoftGlow(SourceImage, BlurRadius, GlowStrength, BrightnessBoost, BlendMode, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to process.	–
BlurRadius Integer, Optional	Blur radius for the glow diffusion in pixels (1-150).	20
GlowStrength Single, Optional	Blending strength (opacity) of the glow layer (0.0-1.0).	0.45F
BrightnessBoost Integer, Optional	Additional brightness boost applied to the final blended pixels (-100 to 100).	5
BlendMode Integer, Optional	Integer representing the blending algorithm used to combine the glow layer with the base image. Supported values: 0: Screen (Bright & Airy) 1: SoftLight (Gentle Contrast)	0
IsShowProgress Boolean, Optional	Whether to show processing progress.	True
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Processed Bitmap with Soft Glow applied, or Nothing if processing fails.

EXAMPLE CODE

```
let glowing = ImgProc.SoftGlow(bitmap, 20, 0.45F, 5, 0);
```

78. ImgProc.StackBlur(SourceImage, BoxSizeHalf, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Applies high-speed stack blur filter.

SYNTAX

```
let result = ImgProc.StackBlur(SourceImage, BoxSizeHalf, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to blur.	–
BoxSizeHalf Integer	Blur radius (half-size).	–
MaskImage Bitmap, Optional	Optional mask to constrain blur.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Blurred image, or Nothing if processing fails.

EXAMPLE CODE

```
let blurred = ImgProc.StackBlur(bitmap, 5);
```

79. ImgProc.StandardFilter(SourceImage, FilterMask(,), WeightFactor, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

Returns: Bitmap

DESCRIPTION

Applies custom convolution filter mask to image.

SYNTAX

```
let result = ImgProc.StandardFilter(SourceImage, FilterMask(,), WeightFactor, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to filter.	–
FilterMask(,) Single		–
WeightFactor Integer, Optional	Weight factor for filter (0 for default).	0
MaskImage Bitmap, Optional	Optional mask to constrain filtering.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Filtered image, or Nothing if processing fails.

EXAMPLE CODE

```
let filtered = ImgProc.StandardFilter(bitmap, {{0,1,0},{1,-4,1},{0,1,0}});
```

80. ImgProc.TextAnnotation(SourceImage, x, y, textString, fontName, fontSize, fontStyleInt, innerColorArgb, outerColorArgb, opacityPercent, outlineThickness, angle, isAntiAlias)

Returns: Bitmap

DESCRIPTION

DESCRIPTION

Annotates image with text and outline style.

SYNTAX

```
let result = ImgProc.TextAnnotation(SourceImage, x, y, textStrin
g, fontName, fontSize, fontStyleInt, innerColorArgb, outerColorAr
gb, opacityPercent, outlineThickness, angle, isAntiAlias);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to annotate.	–
x Single	Text X position.	–
y Single	Text Y position.	–
textString String	Text content to draw.	–
fontName String, Optional	Font name.	"Microsoft Sans Serif"
fontSize Single, Optional	Font size in points.	70.0F
fontStyleInt Integer, Optional	Integer representing the font style. Supported fontStyleInt values: 0: Regular 1: Bold 2: Italic 3: Bold Italic 4: Underline 8: Strikeout	0
innerColorArgb Integer, Optional	Inner text color ARGB.	-1
outerColorArgb Integer, Optional	Outline color ARGB.	-16777216
opacityPercent Integer, Optional	Opacity percentage (0-100).	100
outlineThickness	Outline thickness in pixels.	10.0F

Single, Optional		
angle Single, Optional	Text rotation angle in degrees.	0.0F
isAntiAlias Boolean, Optional	Whether to use anti-aliasing.	True

RETURN VALUE

Bitmap — Annotated image, or Nothing if annotation fails.

EXAMPLE CODE

```
let annotated = ImgProc.TextAnnotation(bitmap, 10, 10, "Hello",
"Arial", 70, 0, -1, -16777216);
```

Processing & Morphological Actions 21 functions

1. ImgProc.BinaryOperation_AND(SourceGrayImage1, SourceGrayImage2, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Performs binary AND operation on two grayscale images.

SYNTAX

```
let result = ImgProc.BinaryOperation_AND(SourceGrayImage1, Source
GrayImage2, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelecte
d);
```

PARAMETERS

Parameter	Description	Default
SourceGrayImage1 Bitmap	First grayscale image. Type: Bitmap; required; passed by reference.	–
SourceGrayImage2 Bitmap	Second grayscale image. Type: Bitmap; required; passed by reference.	–
MaskImage	Optional mask to constrain	Nothing

Bitmap, Optional	operation.	
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Result of AND operation, or Nothing if operation fails.

EXAMPLE CODE

```
let andResult = ImgProc.BinaryOperation_AND(grayBitmap1,
grayBitmap2);
```

2. ImgProc.BinaryOperation_OR(SourceGrayImage1, SourceGrayImage2, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Performs binary OR operation on two grayscale images.

SYNTAX

```
let result = ImgProc.BinaryOperation_OR(SourceGrayImage1, SourceGrayImage2, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceGrayImage1 Bitmap	First grayscale image. Type: Bitmap; required; passed by reference.	–
SourceGrayImage2 Bitmap	Second grayscale image. Type: Bitmap; required; passed by reference.	–
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing

IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Result of OR operation, or Nothing if operation fails.

EXAMPLE CODE

```
let orResult = ImgProc.BinaryOperation_OR(grayBitmap1,
grayBitmap2);
```

3. ImgProc.BinaryOperation_XOR(SourceGrayImage1, SourceGrayImage2, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Performs binary XOR operation on two grayscale images.

SYNTAX

```
let result = ImgProc.BinaryOperation_XOR(SourceGrayImage1, Source
GrayImage2, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelecte
d);
```

PARAMETERS

Parameter	Description	Default
SourceGrayImage1 Bitmap	First grayscale image. Type: Bitmap; required; passed by reference.	–
SourceGrayImage2 Bitmap	Second grayscale image. Type: Bitmap; required; passed by reference.	–
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

`Bitmap` — Result of XOR operation, or Nothing if operation fails.

EXAMPLE CODE

```
let xorResult = ImgProc.BinaryOperation_XOR(grayBitmap1,
grayBitmap2);
```

4. ImgProc.BWLeveling_Using_FloodFillWithTolerance(SourceGrayImage, IsConductBlurring, IsFindThresholdValue, DefaultThresholdValue, Return_OptimalThresholdValue)

Returns: Bitmap

DESCRIPTION

Applies black-white leveling with optional auto thresholding.

SYNTAX

```
let result = ImgProc.BWLeveling_Using_FloodFillWithTolerance(Sour
ceGrayImage, IsConductBlurring, IsFindThresholdValue, DefaultThre
sholdValue, Return_OptimalThresholdValue);
```

PARAMETERS

Parameter	Description	Default
SourceGrayImage Bitmap	Source grayscale image.	–
IsConductBlurring Boolean	Whether to conduct blurring.	–
IsFindThresholdValue Boolean	Whether to auto-find threshold.	–
DefaultThresholdValue Integer	Default threshold value if not auto-finding.	–
Return_OptimalThresholdValue Integer	Output optimal threshold value. Type: Integer; required; passed by reference.	–

RETURN VALUE

`Bitmap` — Leveled black-white image, or Nothing if leveling fails.

EXAMPLE CODE

```
let bwImage =
ImgProc.BWLeveling_Using_FloodFillWithTolerance(grayBitmap, true,
true, 128, optThreshold);
```

5. ImgProc.BWLeveling_Using_FloodFillWithTolerance_Core(SourceGrayImage, ExcludeGrayRange_Min, ExcludeGrayRange_Max)

Returns: Bitmap

DESCRIPTION

Applies black-white leveling excluding specified gray range.

SYNTAX

```
let result = ImgProc.BWLeveling_Using_FloodFillWithTolerance_Core
(SourceGrayImage, ExcludeGrayRange_Min, ExcludeGrayRange_Max);
```

PARAMETERS

Parameter	Description	Default
SourceGrayImage Bitmap	Source grayscale image.	–
ExcludeGrayRange_Min Integer	Minimum gray value to exclude.	–
ExcludeGrayRange_Max Integer	Maximum gray value to exclude.	–

RETURN VALUE

`Bitmap` — Leveled black-white image, or Nothing if leveling fails.

EXAMPLE CODE

```
let bwImage =
ImgProc.BWLeveling_Using_FloodFillWithTolerance_Core(grayBitmap,
100, 150);
```

6. ImgProc.Check_If_CloseShape(Src_BitMap(,), FalseTolerance)

Returns: Boolean

DESCRIPTION

Checks if shape is closed (no gaps).

SYNTAX

```
let result = ImgProc.Check_If_CloseShape(Src_BitMap(,), FalseTolerance);
```

PARAMETERS

Parameter	Description	Default
`Src_BitMap(,)` Boolean		–
`FalseTolerance` Integer, Optional	Tolerance for false detection.	2

RETURN VALUE

Boolean — True if shape is closed, False otherwise.

EXAMPLE CODE

```
let isClosed = ImgProc.Check_If_CloseShape(boolMap, 2);
```

7. ImgProc.Dilate(SourceImage, BoxSizeHalf, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Applies dilation morphological operation to expand bright regions.

SYNTAX

```
let result = ImgProc.Dilate(SourceImage, BoxSizeHalf, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
`SourceImage` Bitmap	Source image to dilate.	–
`BoxSizeHalf` Integer	Half-size of dilation kernel.	–
`MaskImage` Bitmap, Optional	Optional mask to constrain dilation.	Nothing

MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Dilated image, or Nothing if processing fails.

EXAMPLE CODE

```
let dilated = ImgProc.Dilate(bitmap, 1);
```

8. ImgProc.Erode(SourceImage, BoxSizeHalf, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Applies erosion morphological operation to reduce bright regions.

SYNTAX

```
let result = ImgProc.Erode(SourceImage, BoxSizeHalf, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to erode.	–
BoxSizeHalf Integer	Half-size of erosion kernel.	–
MaskImage Bitmap, Optional	Optional mask to constrain erosion.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Eroded image, or Nothing if processing fails.

EXAMPLE CODE

```
let eroded = ImgProc.Erode(bitmap, 1);
```

9. ImgProc.Find_OptimalThreshold_For_BWLeveling_Using_FloodFillWithToler Return_BWGrayImage)

Returns: Integer

DESCRIPTION

Finds optimal threshold for black-white leveling using flood fill.

SYNTAX

```
let result = ImgProc.Find_OptimalThreshold_For_BWLeveling_Using_F
loodFillWithTolerance(SourceGrayImage, Return_BWGrayImage);
```

PARAMETERS

Parameter	Description	Default
`SourceGrayImage` Bitmap	Source grayscale image.	–
`Return_BWGrayImage` Bitmap	Output binary image. Type: Bitmap; required; passed by reference.	–

RETURN VALUE

Integer — Optimal threshold value, or 0 if finding fails.

EXAMPLE CODE

```
let threshold =
ImgProc.Find_OptimalThreshold_For_BWLeveling_Using_FloodFillWithTol
erance(grayBitmap, bwImage);
```

10. ImgProc.RegionColoring(SourceImage, Is8Connected, ApplyBinarization, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Applies region coloring (blob coloring) to the image using a high-speed parallel algorithm.

SYNTAX

```
let result = ImgProc.RegionColoring(SourceImage, Is8Connected, ApplyBinarization, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to color regions.	–
Is8Connected Boolean	Whether to use 8-connectivity.	–
ApplyBinarization Boolean, Optional	Whether to apply binarization.	False
IsShowProgress Boolean, Optional	Whether to show processing progress.	True
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Image with colored regions, or Nothing if operation fails.

EXAMPLE CODE

```
let colored = ImgProc.RegionColoring(bitmap, true, false);
```

11. ImgProc.RegionExtract_ColorFloodFill4(SourceImage, LabelIDMap, RegionColorTable, BoundaryColor, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Extracts color-based connected regions using 4-connectivity flood fill.

SYNTAX

```
let result = ImgProc.RegionExtract_ColorFloodFill4(SourceImage, LabelIDMap, RegionColorTable, BoundaryColor, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image for region extraction.	–
LabelIDMap Integer(,)	Output 2D array of region labels.	–
RegionColorTable Byte(,)	Output table of region colors.	–
BoundaryColor Integer, Optional	Boundary color integer.	16776960
MaskImage Bitmap, Optional	Optional mask to constrain extraction.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

`Bitmap` — Image with color-labeled regions, or Nothing if extraction fails.

EXAMPLE CODE

```
let colorRegions = ImgProc.RegionExtract_ColorFloodFill4(bitmap, labels, colors);
```

12. ImgProc.RegionExtract_ColorFloodFill8(SourceImage, LabelIDMap, RegionColorTable, BoundaryColor, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Extracts color-based connected regions using 8-connectivity flood fill.

SYNTAX

```
let result = ImgProc.RegionExtract_ColorFloodFill8(SourceImage, LabelIDMap, RegionColorTable, BoundaryColor, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image for region extraction.	–
LabelIDMap Integer(,)	Output 2D array of region labels.	–
RegionColorTable Byte(,)	Output table of region colors.	–
BoundaryColor Integer, Optional	Boundary color integer.	16776960
MaskImage Bitmap, Optional	Optional mask to constrain extraction.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

`Bitmap` — Image with color-labeled regions, or Nothing if extraction fails.

EXAMPLE CODE

```
let colorRegions = ImgProc.RegionExtract_ColorFloodFill8(bitmap, labels, colors);
```

13. ImgProc.RegionExtract_FloodFill4(SourceImage, LabelIDMap, RegionColorTable, BkColor, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Extracts connected regions using 4-connectivity flood fill with label mapping.

SYNTAX

```
let result = ImgProc.RegionExtract_FloodFill4(SourceImage, LabelIDMap, RegionColorTable, BkColor, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image for region extraction.	–
LabelIDMap Integer(,)	Output 2D array of region labels.	–
RegionColorTable Byte(,)	Output table of region colors.	–
BkColor Integer, Optional	Background color to identify.	0
MaskImage Bitmap, Optional	Optional mask to constrain extraction.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

`Bitmap` — Image with labeled regions, or Nothing if extraction fails.

EXAMPLE CODE

```
let regionMap = ImgProc.RegionExtract_FloodFill4(bitmap, labels, colors);
```

14. ImgProc.RegionExtract_FloodFill8(SourceImage, LabelIDMap, RegionColorTable, BkColor, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Extracts connected regions using 8-connectivity flood fill with label mapping.

SYNTAX

```
let result = ImgProc.RegionExtract_FloodFill8(SourceImage, LabelIDMap, RegionColorTable, BkColor, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image for region extraction.	–
LabelIDMap Integer(,)	Output 2D array of region labels.	–
RegionColorTable Byte(,)	Output table of region colors.	–
BkColor Integer, Optional	Background color to identify.	0
MaskImage Bitmap, Optional	Optional mask to constrain extraction.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

`Bitmap` — Image with labeled regions, or Nothing if extraction fails.

EXAMPLE CODE

```
let regionMap = ImgProc.RegionExtract_FloodFill8(bitmap, labels, colors);
```

15. ImgProc.RegionExtract_RasterScanning(SourceImage, LabelIDMap, RegionColorTable, BkColor, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Extracts regions using raster scanning algorithm with optional mask support.

SYNTAX

```
let result = ImgProc.RegionExtract_RasterScanning(SourceImage, LabelIDMap, RegionColorTable, BkColor, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image for region extraction.	–
LabelIDMap Integer(,)	Output 2D array of region labels.	–
RegionColorTable Byte(,)	Output table of region colors.	–
BkColor Integer, Optional	Background color to identify.	0
MaskImage Bitmap, Optional	Optional mask to constrain extraction.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

`Bitmap` — Image with labeled regions, or Nothing if extraction fails.

EXAMPLE CODE

```
let regions = ImgProc.RegionExtract_RasterScanning(bitmap, labels, colors);
```

16. ImgProc.RegionExtract_RasterScanning_NoMaskImage(SourceImage, LabelIDMap, RegionColorTable, BkColor)

Returns: Bitmap

DESCRIPTION

Extracts regions using raster scanning algorithm without mask image.

SYNTAX

```
let result = ImgProc.RegionExtract_RasterScanning_NoMaskImage(Sou
rceImage, LabelIDMap, RegionColorTable, BkColor);
```

PARAMETERS

Parameter	Description	Default
`SourceImage` Bitmap	Source image for region extraction.	–
`LabelIDMap` Integer(,)	Output 2D array of region labels.	–
`RegionColorTable` Byte(,)	Output table of region colors.	–
`BkColor` Integer, Optional	Background color to identify.	0

RETURN VALUE

`Bitmap` — Image with labeled regions, or Nothing if extraction fails.

EXAMPLE CODE

```
let regions =
ImgProc.RegionExtract_RasterScanning_NoMaskImage(bitmap, labels,
colors);
```

17. ImgProc.RemoveDots(SourceImage, DotMode, Is8Connected, MaxDotSize, ApplyBinarization, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Removes small isolated dots or noise from the image using parallel processing.

SYNTAX

```
let result = ImgProc.RemoveDots(SourceImage, DotMode, Is8Connecte
d, MaxDotSize, ApplyBinarization, IsShowProgress, MaskImage, Mask
RegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
`SourceImage` Bitmap	Source image to process.	–

DotMode Integer	Dot removal mode. Supported values: 0: Remove black dots (filled with white) 1: Remove white dots (filled with black) 2: Remove both black and white dots (filled with the opposite color)	–
Is8Connected Boolean	Whether to use 8-connectivity.	–
MaxDotSize Integer	Maximum size of dots to remove.	–
ApplyBinarization Boolean, Optional	Whether to apply binarization.	`True`
IsShowProgress Boolean, Optional	Whether to show processing progress.	`True`
MaskImage Bitmap, Optional	Optional mask to constrain operation.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Bitmap` — Processed image with dots removed, or Nothing if operation fails.

EXAMPLE CODE

```
let cleaned = ImgProc.RemoveDots(bitmap, 0, true, 10, true);
```

18. ImgProc.Skeletonize(SourceImage, IsExtended, IsInverted, MinBranchLength, TailLength, ApplyBinarization, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Reduces binary objects to 1-pixel wide skeletons using parallel processing.

SYNTAX

```
let result = ImgProc.Skeletonize(SourceImage, IsExtended, IsInverted, MinBranchLength, TailLength, ApplyBinarization, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to skeletonize.	–
IsExtended Boolean	Whether to use extended skeletonization.	–
IsInverted Boolean	Whether to invert the image before processing.	–
MinBranchLength Integer	Minimum branch length in pixels.	–
TailLength Integer	Tail length for pruning.	–
ApplyBinarization Boolean, Optional	Whether to apply binarization.	`True`
IsShowProgress Boolean, Optional	Whether to show processing progress.	`True`
MaskImage Bitmap, Optional	Optional mask to constrain operation.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Bitmap` — Skeletonized image, or Nothing if operation fails.

EXAMPLE CODE

```
let skeleton = ImgProc.Skeletonize(bitmap, true, false, 10, 5, true);
```

19. ImgProc.SplitObjects_ConvexityDefects(SourceImage, MinDefectDepth, MaxCutLength, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Splits connected objects using convexity defects.

SYNTAX

```
let result = ImgProc.SplitObjects_ConvexityDefects(SourceImage, MinDefectDepth, MaxCutLength, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image with connected objects.	–
MinDefectDepth Single, Optional	Minimum defect depth threshold.	5.0F
MaxCutLength Single, Optional	Maximum cut length for splitting.	50.0F
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Image with split objects, or Nothing if splitting fails.

EXAMPLE CODE

```
let split = ImgProc.SplitObjects_ConvexityDefects(bitmap, 5.0, 50.0);
```

20. ImgProc.SuperPixelSegmentation(SourceImage, numSuperpixels, compactness, fillWithWhite, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Segments an image into superpixels using parallel SLIC algorithm.

SYNTAX

```
let result = ImgProc.SuperPixelSegmentation(SourceImage, numSuperpixels, compactness, fillWithWhite, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to segment.	–
numSuperpixels Integer, Optional	Target number of superpixels.	400
compactness Single, Optional	Spatial weight for compactness.	70.0F
fillWithWhite Boolean, Optional	Whether to fill background with white.	False
MaskImage Bitmap, Optional	Optional mask to constrain segmentation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

`Bitmap` — Segmented image with superpixel boundaries, or Nothing if processing fails.

EXAMPLE CODE

```
let segmented = ImgProc.SuperPixelSegmentation(bitmap, 400, 70.0, false);
```

21. ImgProc.WatershedSegmentation(SourceImage, MinDistanceThreshold, SmoothRadius, BoundaryThickness, BoundR,

BoundG, BoundB, ReturnOnlyBoundaries, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Segments image using watershed algorithm.

SYNTAX

```
let result = ImgProc.WatershedSegmentation(SourceImage, MinDistanceThreshold, SmoothRadius, BoundaryThickness, BoundR, BoundG, BoundB, ReturnOnlyBoundaries, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to segment.	–
MinDistanceThreshold Integer, Optional	Minimum distance threshold.	10
SmoothRadius Integer, Optional	Smoothing radius.	4
BoundaryThickness Integer, Optional	Boundary line thickness.	1
BoundR Byte, Optional	Red channel for boundary color.	0
BoundG Byte, Optional	Green channel for boundary color.	0
BoundB Byte, Optional	Blue channel for boundary color.	0
ReturnOnlyBoundaries Boolean, Optional	Whether to return only boundary lines.	True
MaskImage Bitmap, Optional	Optional mask to constrain segmentation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected	Whether entire region is	True

	Boolean, Optional	selected.

RETURN VALUE

`Bitmap` — Segmented image, or Nothing if segmentation fails.

EXAMPLE CODE

```
let segmented = ImgProc.WatershedSegmentation(bitmap, 10, 4, 1);
```

System & Workspace Actions 26 functions

1. ImgProc.CaptureDesktopScreen(PrintCursor) Returns: Bitmap

DESCRIPTION

Captures entire desktop screen to bitmap.

SYNTAX

```
let result = ImgProc.CaptureDesktopScreen(PrintCursor);
```

PARAMETERS

Parameter	Description	Default
`PrintCursor` Boolean, Optional	Whether to render cursor on captured screen.	`True`

RETURN VALUE

`Bitmap` — Captured desktop screen image, or Nothing if capture fails.

EXAMPLE CODE

```
let screenshot = ImgProc.CaptureDesktopScreen();
```

2. ImgProc.CaptureDesktopScreenByRegion(PrintCursor, IsAllRegion, X1, Y1, X2, Y2)

Returns: Bitmap

DESCRIPTION

Captures specified region of desktop screen.

SYNTAX

```
let result = ImgProc.CaptureDesktopScreenByRegion(PrintCursor, Is
```

```
AllRegion, X1, Y1, X2, Y2);
```

PARAMETERS

Parameter	Description	Default
`PrintCursor` Boolean	Whether to render cursor on captured region.	–
`IsAllRegion` Boolean	Whether to capture entire screen region.	–
`X1` Integer	Left coordinate of region.	–
`Y1` Integer	Top coordinate of region.	–
`X2` Integer	Right coordinate of region.	–
`Y2` Integer	Bottom coordinate of region.	–

RETURN VALUE

`Bitmap` — Captured region image, or Nothing if capture fails.

EXAMPLE CODE

```
let regionCapture = ImgProc.CaptureDesktopScreenByRegion(false, false, 0, 0, 800, 600);
```

3. ImgProc.CaptureObject(pHwnd) Returns: Bitmap

DESCRIPTION

Captures window or UI object to bitmap.

SYNTAX

```
let result = ImgProc.CaptureObject(pHwnd);
```

PARAMETERS

Parameter	Description	Default
`pHwnd` IntPtr	Handle to window or UI object.	–

RETURN VALUE

`Bitmap` — Captured control image, or Nothing if capture fails.

EXAMPLE CODE

```
let windowCapture = ImgProc.CaptureObject(formHandle);
```

4. ImgProc.ChromaKey(TargetCanvas, OverlayImage, ChromaR, ChromaG, ChromaB, Opacity, IsFeather, FeatherRadius, ChromaTolerance, ChromaSoftness, extractTransparent, MaskImage, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Applies chroma key (green screen) effect to composite images.

SYNTAX

```
let result = ImgProc.ChromaKey(TargetCanvas, OverlayImage, ChromaR, ChromaG, ChromaB, Opacity, IsFeather, FeatherRadius, ChromaTolerance, ChromaSoftness, extractTransparent, MaskImage, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
TargetCanvas Bitmap	Target canvas background image.	–
OverlayImage Bitmap	Foreground image to composite.	–
ChromaR Byte	Red component of chroma key color (0-255).	–
ChromaG Byte	Green component of chroma key color (0-255).	–
ChromaB Byte	Blue component of chroma key color (0-255).	–
Opacity Single	Opacity factor (0.0-1.0).	–
IsFeather Boolean	Whether to feather edges.	–

FeatherRadius Integer	Feathering radius in pixels.	–
ChromaTolerance Single, Optional	Color matching tolerance range (0.0-100.0).	35.0F
ChromaSoftness Single, Optional	Edge gradient softness range (0.0-100.0).	20.0F
extractTransparent Boolean, Optional	Whether to extract transparency.	False
MaskImage Bitmap, Optional	Optional mask to constrain effect.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Composited image with chroma key applied, or Nothing if compositing fails.

EXAMPLE CODE

```
let composite = ImgProc.ChromaKey(canvas, overlay, 0, 255, 0, 1.0F, true, 5, 35.0F, 20.0F);
```

5. ImgProc.CloneImage(SourceImage) Returns: Bitmap

DESCRIPTION

Creates a deep copy of the image.

SYNTAX

```
let result = ImgProc.CloneImage(SourceImage);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to clone.	–

RETURN VALUE

Bitmap — Independent clone of image, or Nothing if cloning fails.

EXAMPLE CODE

```
let clone = ImgProc.CloneImage(bitmap);
```

6. ImgProc.CopyToClipboardWithTransparency(img) Returns: Void

DESCRIPTION

Copies image to clipboard with transparency.

SYNTAX

```
ImgProc.CopyToClipboardWithTransparency(img);
```

PARAMETERS

Parameter	Description	Default
img Bitmap	Image to copy.	–

RETURN VALUE

Void — None.

EXAMPLE CODE

```
ImgProc.CopyToClipboardWithTransparency(bitmap);
```

7. ImgProc.Draw_ColorLevelBox(ColorMode, Pic_Target, LevelCount)

Returns: Void

DESCRIPTION

Draws color level distribution box in picture control.

SYNTAX

```
ImgProc.Draw_ColorLevelBox(ColorMode, Pic_Target, LevelCount);
```

PARAMETERS

Parameter	Description	Default
ColorMode Integer	Color mode for the distribution. Supported values: 1: Red 2: Green 3: Blue	–
Pic_Target	Target picture control to draw in	

Pic_Target PictureBox	Target picture control to draw in. Type: PictureBox; required; passed by reference.	–
LevelCount Integer	Number of discrete levels to display.	–

RETURN VALUE

Void

EXAMPLE CODE

```
ImgProc.Draw_ColorLevelBox(0, picTarget, 8);
```

8. ImgProc.Draw_Line(Pic_Target, X1, Y1, X2, Y2, LineColor)

Returns: Bitmap

DESCRIPTION

Draws line on picture control.

SYNTAX

```
let result = ImgProc.Draw_Line(Pic_Target, X1, Y1, X2, Y2, LineColor);
```

PARAMETERS

Parameter	Description	Default
Pic_Target PictureBox	Target picture control. Type: PictureBox; required; passed by reference.	–
X1 Integer	Starting X-coordinate.	–
Y1 Integer	Starting Y-coordinate.	–
X2 Integer	Ending X-coordinate.	–
Y2 Integer	Ending Y-coordinate.	–
LineColor Color	Color of the line.	–

RETURN VALUE

`Bitmap` — Bitmap with drawn line, or Nothing if drawing fails.

EXAMPLE CODE

```
let lineBmp = ImgProc.Draw_Line(picTarget, 0, 0, 100, 100,
Color.Red);
```

9. ImgProc.Draw_Text_Outlined(Pic_Target, x, y, TextString, TextFont, InnerColor, OuterColor)

Returns: Bitmap

DESCRIPTION

Draws text with outline in picture control.

SYNTAX

```
let result = ImgProc.Draw_Text_Outlined(Pic_Target, x, y, TextStr
ing, TextFont, InnerColor, OuterColor);
```

PARAMETERS

Parameter	Description	Default
`Pic_Target` PictureBox	Target picture control. Type: PictureBox; required; passed by reference.	–
`x` Integer	X-coordinate for text.	–
`y` Integer	Y-coordinate for text.	–
`TextString` String	Text string to draw.	–
`TextFont` Font	Font for text.	–
`InnerColor` Color	Inner text color.	–
`OuterColor` Color	Outline color.	–

RETURN VALUE

Bitmap — Bitmap containing drawn text, or Nothing if drawing fails.

EXAMPLE CODE

```
let textBmp = ImgProc.Draw_Text_Outlined(picTarget, 10, 10,
"Hello", font, Color.White, Color.Black);
```

10. ImgProc.Draw_Text_Outlined_Bitmap(Pic_Target, x, y, TextString, TextFont, InnerColor, OuterColor)

Returns: Bitmap

DESCRIPTION

Draws outlined text on bitmap.

SYNTAX

```
let result = ImgProc.Draw_Text_Outlined_Bitmap(Pic_Target, x, y,
TextString, TextFont, InnerColor, OuterColor);
```

PARAMETERS

Parameter	Description	Default
Pic_Target Bitmap	Target bitmap to draw text on.	–
x Integer	X-coordinate for text.	–
y Integer	Y-coordinate for text.	–
TextString String	Text string to draw.	–
TextFont Font	Font for text.	–
InnerColor Color	Inner text color.	–
OuterColor Color	Outline color.	–

RETURN VALUE

Bitmap — Bitmap with drawn text, or Nothing if drawing fails.

EXAMPLE CODE

```
let textBmp = ImgProc.Draw_Text_Outlined_Bitmap(bitmap, 10, 10,
"Hello", font, Color.White, Color.Black);
```

11. ImgProc.Draw_Text_Outlined_Graphics(SourceGraphics, x, y, TextString, TextFont, InnerColor, OuterColor)

Returns: Void

DESCRIPTION

Draws outlined text on graphics object.

SYNTAX

```
ImgProc.Draw_Text_Outlined_Graphics(SourceGraphics, x, y, TextStr
ing, TextFont, InnerColor, OuterColor);
```

PARAMETERS

Parameter	Description	Default
SourceGraphics Graphics	Target graphics object to draw on.	–
x Integer	X-coordinate for text.	–
y Integer	Y-coordinate for text.	–
TextString String	Text string to draw.	–
TextFont Font	Font for text.	–
InnerColor Color	Inner text color.	–
OuterColor Color	Outline color.	–

RETURN VALUE

Void

EXAMPLE CODE

```
ImgProc.Draw_Text_Outlined_Graphics(g, 10, 10, "Hello", font,
```

```
Color.White, Color.Black);
```

12. ImgProc.Draw_Text_Outlined_Image(SourceImage, x, y, TextString, TextFont, InnerColor, OuterColor)

Returns: Bitmap

DESCRIPTION

Draws outlined text on image.

SYNTAX

```
let result = ImgProc.Draw_Text_Outlined_Image(SourceImage, x, y, TextString, TextFont, InnerColor, OuterColor);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to draw text on.	–
x Integer	X-coordinate for text.	–
y Integer	Y-coordinate for text.	–
TextString String	Text string to draw.	–
TextFont Font	Font for text.	–
InnerColor Color	Inner text color.	–
OuterColor Color	Outline color.	–

RETURN VALUE

`Bitmap` — Image with drawn text, or Nothing if drawing fails.

EXAMPLE CODE

```
let textImg = ImgProc.Draw_Text_Outlined_Image(bitmap, 10, 10, "Hello", font, Color.White, Color.Black);
```

13. ImgProc.DrawSelectionBorder(SourceImage, cR, cG, cB, BorderThickness, Opacity)

Returns: Bitmap

DESCRIPTION

Draws selection border with RGB values on image.

SYNTAX

```
let result = ImgProc.DrawSelectionBorder(SourceImage, cR, cG, cB, BorderThickness, Opacity);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to draw on.	–
cR Byte	Red component (0-255).	–
cG Byte	Green component (0-255).	–
cB Byte	Blue component (0-255).	–
BorderThickness Integer	Thickness of border line in pixels.	–
Opacity Byte	Alpha transparency (0-255).	–

RETURN VALUE

`Bitmap` — Image with selection border drawn, or Nothing if drawing fails.

EXAMPLE CODE

```
let bordered = ImgProc.DrawSelectionBorder(bitmap, 255, 0, 0, 2, 255);
```

14. ImgProc.FillRegion(SourceImage, FillColorRed, FillColorGreen, FillColorBlue, IsFillInside, Opacity, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Fills image regions with RGB color value and opacity.

SYNTAX

```
let result = ImgProc.FillRegion(SourceImage, FillColorRed, FillColorGreen, FillColorBlue, IsFillInside, Opacity, IsShowProgress, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source image to fill.	–
FillColorRed Byte	Red fill color (0-255).	–
FillColorGreen Byte	Green fill color (0-255).	–
FillColorBlue Byte	Blue fill color (0-255).	–
IsFillInside Boolean	Whether to fill inside (True) or outside (False).	–
Opacity Single	Fill opacity (0-1).	–
IsShowProgress Boolean, Optional	Whether to display progress.	`True`
MaskImage Bitmap, Optional	Optional mask to constrain filling.	`Nothing`
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	`Nothing`
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	`True`

RETURN VALUE

`Bitmap` — Filled image, or Nothing if filling fails.

EXAMPLE CODE

```
let filled = ImgProc.FillRegion(bitmap, 255, 0, 0, true, 0.5);
```

15. ImgProc.FloodFill(SourceImage, CenterX, CenterY, FillColorR, FillColorG, FillColorB, ToleranceR, ToleranceG, ToleranceB, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: Bitmap

DESCRIPTION

Fills region starting from center point with color.

SYNTAX

```
let result = ImgProc.FloodFill(SourceImage, CenterX, CenterY, FillColorR, FillColorG, FillColorB, ToleranceR, ToleranceG, ToleranceB, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
`SourceImage` Bitmap	Source image to fill.	–
`CenterX` Integer	Center X-coordinate to start fill.	–
`CenterY` Integer	Center Y-coordinate to start fill.	–
`FillColorR` Byte	Fill red component.	–
`FillColorG` Byte	Fill green component.	–
`FillColorB` Byte	Fill blue component.	–
`ToleranceR` Integer	Red color tolerance.	–
`ToleranceG` Integer	Green color tolerance.	–
`ToleranceB` Integer	Blue color tolerance.	–

MaskImage Bitmap, Optional	Optional mask to constrain fill.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

Bitmap — Filled image, or Nothing if fill fails.

EXAMPLE CODE

```
let filled = ImgProc.FloodFill(bitmap, 50, 50, 255, 0, 0, 10, 10, 10);
```

16. ImgProc.Generate_BlankImage(Pic_Target, BackColor)

Returns: Void

DESCRIPTION

Generates blank image in picture control.

SYNTAX

```
ImgProc.Generate_BlankImage(Pic_Target, BackColor);
```

PARAMETERS

Parameter	Description	Default
Pic_Target PictureBox	Target picture control. Type: PictureBox; required; passed by reference.	–
BackColor Color	Background color for the blank image.	–

RETURN VALUE

Void

EXAMPLE CODE

```
ImgProc.Generate_BlankImage(picTarget, Color.White);
```

17. ImgProc.Get_ActualFormRect(SrcForm) Returns: Rectangle

DESCRIPTION

Gets actual rectangle bounds of form.

SYNTAX

```
let result = ImgProc.Get_ActualFormRect(SrcForm);
```

PARAMETERS

Parameter	Description	Default
`SrcForm` Form	Source form window.	–

RETURN VALUE

Rectangle — Rectangle bounds of the form, or Rectangle.Empty if retrieval fails.

EXAMPLE CODE

```
let bounds = ImgProc.Get_ActualFormRect(myForm);
```

18. ImgProc.Get_CoreCount() Returns: Integer

DESCRIPTION

Gets number of processor cores.

SYNTAX

```
let result = ImgProc.Get_CoreCount();
```

RETURN VALUE

Integer — Number of available processor cores, or 1 if retrieval fails.

EXAMPLE CODE

```
let cores = ImgProc.Get_CoreCount();
```

19. ImgProc.Get_CursorPosition(x, y) Returns: Void

DESCRIPTION

Gets current cursor position on screen.

SYNTAX

```
ImgProc.Get_CursorPosition(x, y);
```

PARAMETERS

Parameter	Description	Default
`x` Integer	Output cursor X coordinate. Type: Integer; required; passed by reference.	–
`y` Integer	Output cursor Y coordinate. Type: Integer; required; passed by reference.	–

RETURN VALUE

`Void` — None.

EXAMPLE CODE

```
ImgProc.Get_CursorPosition(cursorX, cursorY);
```

20. ImgProc.Get_ImageFormat(FileExtension) Returns: Object

DESCRIPTION

Gets image format object from file extension.

SYNTAX

```
let result = ImgProc.Get_ImageFormat(FileExtension);
```

PARAMETERS

Parameter	Description	Default
`FileExtension` String	File extension (e.g., '.jpg', '.png').	–

RETURN VALUE

`Object` — Image format object, or Nothing if format not found.

EXAMPLE CODE

```
let fmt = ImgProc.Get_ImageFormat(".jpg");
```

21. ImgProc.Get_WindowHeightMargin(TargetForm) Returns: Integer

DESCRIPTION

Gets window height margin value.

SYNTAX

```
let result = ImgProc.Get_WindowHeightMargin(TargetForm);
```

PARAMETERS

Parameter	Description	Default
`TargetForm` Form	Target form window.	–

RETURN VALUE

Integer — Window height margin, or 0 if retrieval fails.

EXAMPLE CODE

```
let margin = ImgProc.Get_WindowHeightMargin(myForm);
```

22. ImgProc.Get_WindowWidthMargin(TargetForm) Returns: Integer

DESCRIPTION

Gets window width margin value.

SYNTAX

```
let result = ImgProc.Get_WindowWidthMargin(TargetForm);
```

PARAMETERS

Parameter	Description	Default
`TargetForm` Form	Target form window.	–

RETURN VALUE

Integer — Window width margin, or 0 if retrieval fails.

EXAMPLE CODE

```
let margin = ImgProc.Get_WindowWidthMargin(myForm);
```

23. ImgProc.GetControlBitmap(CTL) Returns: Bitmap

DESCRIPTION

Gets bitmap snapshot of UI control.

SYNTAX

```
let result = ImgProc.GetControlBitmap(CTL);
```

PARAMETERS

Parameter	Description	Default
CTL Control	Source UI control.	–

RETURN VALUE

`Bitmap` — Bitmap snapshot of the control, or Nothing if snapshot fails.

EXAMPLE CODE

```
let ctlBitmap = ImgProc.GetControlBitmap(myPictureBox);
```

24. ImgProc.SaveDepthMapToSTL_BoxFilter(SourceImage, FilePath, MaxSTLHeight, BaseThickness, SmoothingPasses, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: String

DESCRIPTION

Saves depth map as STL file using box filter.

SYNTAX

```
let result = ImgProc.SaveDepthMapToSTL_BoxFilter(SourceImage, FilePath, MaxSTLHeight, BaseThickness, SmoothingPasses, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source depth map image.	–
FilePath String	Path to save STL file.	–
MaxSTLHeight Single, Optional	Maximum STL height.	15.0F
BaseThickness	Base thickness for STL.	2.0F

Single, Optional		
SmoothingPasses Integer, Optional	Number of smoothing passes.	3
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

String — File path if save successful, empty string if operation fails.

EXAMPLE CODE

```
let result = ImgProc.SaveDepthMapToSTL_BoxFilter(depthBitmap,
"output.stl", 15.0, 2.0, 3);
```

25. ImgProc.SaveDepthMapToSTL_GaussianFilter(SourceImage, FilePath, MaxSTLHeight, BaseThickness, SmoothingPasses, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected)

Returns: String

DESCRIPTION

Saves depth map as STL file using Gaussian filter.

SYNTAX

```
let result = ImgProc.SaveDepthMapToSTL_GaussianFilter(SourceImage, FilePath, MaxSTLHeight, BaseThickness, SmoothingPasses, MaskImage, MaskRegionBoundary, IsMaskAllRegionSelected);
```

PARAMETERS

Parameter	Description	Default
SourceImage Bitmap	Source depth map image.	–
FilePath String	Path to save STL file.	–
MaxSTLHeight	Maximum STL height.	15.0F

Single, Optional		
BaseThickness Single, Optional	Base thickness for STL.	2.0F
SmoothingPasses Integer, Optional	Number of smoothing passes.	3
MaskImage Bitmap, Optional	Optional mask to constrain operation.	Nothing
MaskRegionBoundary Rectangle, Optional	Boundary rectangle for masked region.	Nothing
IsMaskAllRegionSelected Boolean, Optional	Whether entire region is selected.	True

RETURN VALUE

String — File path if save successful, empty string if operation fails.

EXAMPLE CODE

```
let result = ImgProc.SaveDepthMapToSTL_GaussianFilter(depthBitmap,
"output.stl", 15.0, 2.0, 3);
```

26. ImgProc.Stop_Processing() Returns: Void

DESCRIPTION

Stops the current image processing operation.

SYNTAX

```
ImgProc.Stop_Processing();
```

RETURN VALUE

Void

EXAMPLE CODE

```
ImgProc.Stop_Processing();
```

5. ObjectDetector (ScriptObjectDetectorHelper) 8 Functions

Class Overview: Provides robust YOLOv3 ONNX-based object detection. Features hardware-accelerated AI inference, bounding box extraction, and class counting capabilities.

Neural & AI Processing Actions

8 functions

1. ObjectDetector.Detect(img, confThreshold, nmsThreshold, useTiling)

Returns: Integer

DESCRIPTION

Detects objects in an image and stores the detection results.

SYNTAX

```
let result = ObjectDetector.Detect(img, confThreshold, nmsThreshold, useTiling);
```

PARAMETERS

Parameter	Description	Default
img Bitmap	Image to analyze.	–
confThreshold Double, Optional	Minimum confidence threshold.	0.5F
nmsThreshold Double, Optional	Non-maximum suppression threshold.	0.4F
useTiling Boolean, Optional	Whether to use tiled inference.	False

RETURN VALUE

Integer — Number of detected objects, or 0 if detection fails.

EXAMPLE CODE

```
let count = ObjectDetector.Detect(bitmap, 0.5, 0.4, false);
```

2. ObjectDetector.GetAllResults()

Returns: ONNX_YOLOv3Detector.DetectionResult()

DESCRIPTION

Returns all detection results from the most recent detection.

SYNTAX

```
let result = ObjectDetector.GetAllResults();
```

RETURN VALUE

`ONNX_YOLOv3Detector.DetectionResult()` — Array of detection results, or an empty array when no results are available.

EXAMPLE CODE

```
let results = ObjectDetector.GetAllResults();
```

3. ObjectDetector.GetClassCount(className) Returns: Integer

DESCRIPTION

Returns the number of detected objects with the specified class label.

SYNTAX

```
let result = ObjectDetector.GetClassCount(className);
```

PARAMETERS

Parameter	Description	Default
className String	Class label to count.	–

RETURN VALUE

`Integer` — Number of detections for the class.

EXAMPLE CODE

```
let count = ObjectDetector.GetClassCount("person");
```

4. ObjectDetector.GetClassLocations(className) Returns: RectangleF()

DESCRIPTION

Returns bounding boxes for detections of the specified class.

SYNTAX

```
let result = ObjectDetector.GetClassLocations(className);
```

PARAMETERS

Parameter	Description	Default
`className` String	Class label to locate.	–

RETURN VALUE

`RectangleF()` — Array of bounding rectangles for matching detections.

EXAMPLE CODE

```
let locations = ObjectDetector.GetClassLocations("person");
```

5. ObjectDetector.GetLastError() Returns: String

DESCRIPTION

Returns the most recent error message produced by the helper.

SYNTAX

```
let result = ObjectDetector.GetLastError();
```

RETURN VALUE

`String` — The last error message, or an empty string when no error is recorded.

EXAMPLE CODE

```
let errorMessage = ObjectDetector.GetLastError();
```

6. ObjectDetector.GetRenderedImage(img, showConfidence, colorName)

Returns: Bitmap

DESCRIPTION

Returns a copy of the image rendered with detection boxes and labels.

SYNTAX

```
let result = ObjectDetector.GetRenderedImage(img, showConfidence, colorName);
```

PARAMETERS

Parameter	Description	Default

img Bitmap	Source image to render.	–
showConfidence Boolean, Optional	Whether to display confidence values.	True
colorName String, Optional	Known .NET color name for boxes and labels.	"Red"

RETURN VALUE

Bitmap — Rendered bitmap, or Nothing if rendering fails.

EXAMPLE CODE

```
let rendered = ObjectDetector.GetRenderedImage(bitmap, true,
"Lime");
```

7. ObjectDetector.HasClass(className) Returns: Boolean

DESCRIPTION

Checks whether the most recent results contain the specified object class.

SYNTAX

```
let result = ObjectDetector.HasClass(className);
```

PARAMETERS

Parameter	Description	Default
className String	Class label to find.	–

RETURN VALUE

Boolean — True if the class was detected; otherwise False.

EXAMPLE CODE

```
let found = ObjectDetector.HasClass("person");
```

8. ObjectDetector.Initialize() Returns: Boolean

DESCRIPTION

Initializes the YOLOv3 detector with its default model and settings.

SYNTAX

```
let result = ObjectDetector.Initialize();
```

RETURN VALUE

Boolean — True when the detector is initialized or already available; otherwise False.

EXAMPLE CODE

```
let ready = ObjectDetector.Initialize();
```

6. Region (ScriptRegionHelper) 38 Functions

Class Overview: The Region object provides programmatic access for defining, modifying, inspecting, and manipulating ROI (Region of Interest) selections, shape geometries, and binary masks in PhenoCapture.

Edit & Selection Actions 38 functions

1. Region.AddCircle(left, top, right, bottom) Returns: Void

DESCRIPTION

Adds a circular area defined by a bounding rectangle to the current multi-region selection.

SYNTAX

```
Region.AddCircle(left, top, right, bottom);
```

PARAMETERS

Parameter	Description	Default
`left` Integer	Left X coordinate.	–
`top` Integer	Top Y coordinate.	–
`right` Integer	Right X coordinate.	–
`bottom`	Bottom Y coordinate.	–

Integer

RETURN VALUE

`Void` — Void. Coordinates are clipped to the workspace image bounds.

EXAMPLE CODE

```
Region.AddCircle(10, 20, 100, 120);
```

2. Region.AddMaskImage(appendBmp, offsetX, offsetY) Returns: Void

DESCRIPTION

Adds mask pixels from another bitmap at the specified offset.

SYNTAX

```
Region.AddMaskImage(appendBmp, offsetX, offsetY);
```

PARAMETERS

Parameter	Description	Default
appendBmp Bitmap	Bitmap whose white pixels are added to the mask.	–
offsetX Integer, Optional	Horizontal offset in pixels.	0
offsetY Integer, Optional	Vertical offset in pixels.	0

RETURN VALUE

`Void` — Void.

EXAMPLE CODE

```
Region.AddMaskImage(mask, 10, 5);
```

3. Region.AddPolygon(coords) Returns: Void

DESCRIPTION

Adds a polygonal area to the current multi-region selection.

SYNTAX

```
Region.AddPolygon(coords);
```

PARAMETERS

Parameter	Description	Default
`coords` Integer()	At least three coordinate pairs in X, Y order.	–

RETURN VALUE

`Void` — Void. Coordinates are clipped to the workspace image bounds.

EXAMPLE CODE

```
Region.AddPolygon(New Integer() {10, 10, 100, 10, 50, 80});
```

4. Region.AddRectangle(left, top, right, bottom) Returns: Void

DESCRIPTION

Adds a rectangular area to the current multi-region selection.

SYNTAX

```
Region.AddRectangle(left, top, right, bottom);
```

PARAMETERS

Parameter	Description	Default
`left` Integer	Left X coordinate.	–