MIL 8.0 GUIDE

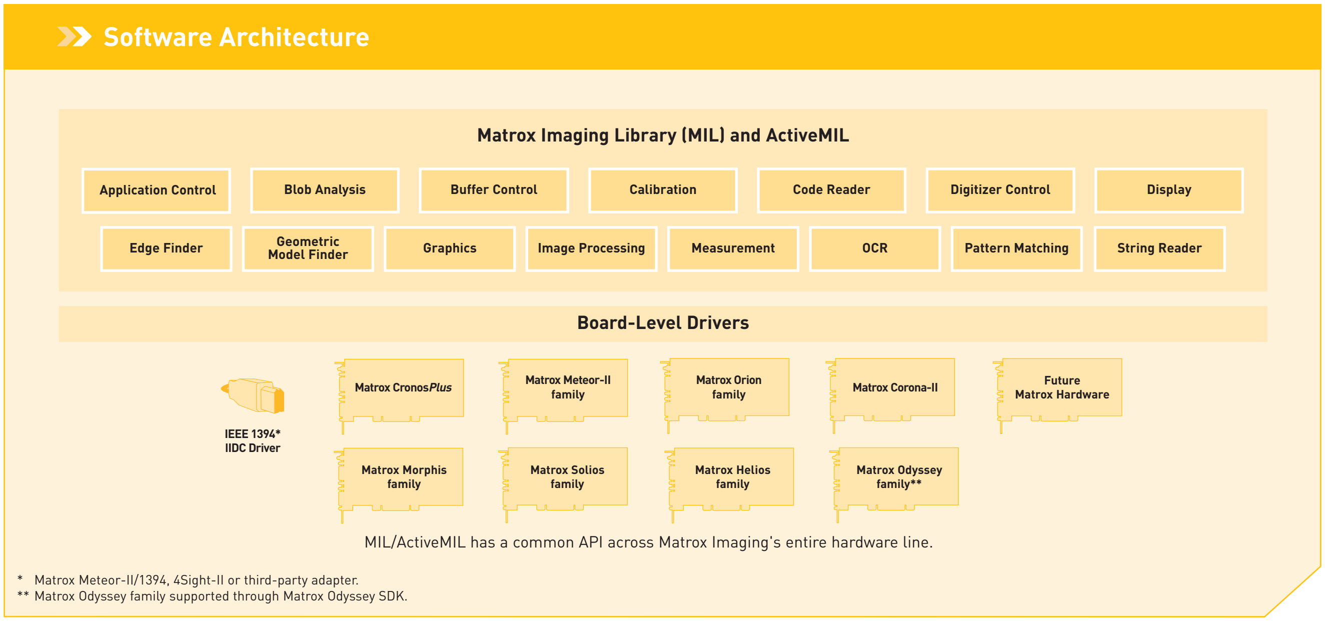
Including **Active MIL**

MATROX
IMAGING

Overview

Matrox Imaging Library (MIL) is a modular programming library with commands for image capture, image processing, pattern recognition, blob analysis, edge extraction and analysis, measurement, character recognition, 1D and 2D code reading, calibration, graphics, image compression, image display and archiving. Included with MIL is ActiveMIL, a collection of ActiveX controls (OCXs) for managing image capture, processing, analysis, display and archiving.

This guide has been designed to complement the Matrox Imaging Library (MIL) brochure by providing a list of benchmarks on different platforms and a comprehensive overview of the MIL and ActiveMIL APIs. Included with the command and parameter descriptions of MIL are real programming examples for each module. Also included are brief descriptions of some of the control methods, events and properties available in ActiveMIL. For additional information on MIL commands and parameters, as well as ActiveMIL control methods, events, and properties, refer to the MIL and ActiveMIL Command Reference respectively.



Software Architecture

Matrox Imaging Library (MIL) and ActiveMIL

Application Control
Blob Analysis
Buffer Control
Calibration
Code Reader
Digitizer Control
Display
Edge Finder
Geometric Model Finder
Graphics
Image Processing
Measurement
OCR
Pattern Matching
String Reader

Board-Level Drivers

IEEE 1394*
IIDC Driver
Matrox Cronos*Plus*
Matrox Meteor-II family
Matrox Orion family
Matrox Corona-II
Future Matrox Hardware
Matrox Morphis family
Matrox Solios family
Matrox Helios family
Matrox Odyssey family**

MIL/ActiveMIL has a common API across Matrox Imaging's entire hardware line.

* Matrox Meteor-II/1394, 4Sight-II or third-party adapter.
** Matrox Odyssey family supported through Matrox Odyssey SDK.

Table of Contents

MIL Command Listing and Description

Programming Examples

ActiveMIL Control Listing and Description

*Available as of Processing Pack 1.

MIL/ActiveMIL Benchmarks

The following benchmarks provide a performance overview for a range of imaging operations running on different platforms. A brief description of all functions, parameters and images used are included. Note that the benchmarks assume full processor and memory bandwidth (i.e., no other system activity), and include command overheads.

Note: Operations executed on 512 x 512 images[1].

Image Processing

	1.3 GHz Celeron™ M 512KB L2 Cache 400 MHz FSB PC2700 SDRAM (Matrox 4Sight M)	2.0 GHz Pentium™ M 2MB L2 Cache 400 MHz FSB PC2700 SDRAM (Matrox 4Sight M)	2.6 GHz Opteron™ 1 MB L2 Cache DDR1-400 SDRAM	3.6 GHz Xeon™ 1 MB L2 Cache 800 MHz FSB DDR2-400 SDRAM	Helios XA / XCL 133 MHz PA (with 3.6 GHz Xeon™)
Point-to-point Add two 8-bit images and store results in an 8-bit destination image.	0.58 ms	0.08 ms	0.08 ms	0.05 ms	0.23 ms[2]
Edge Detection (sobel) Perform an edge detection (sobel) on an 8-bit source image and store results in an 8-bit destination image.	1.4 ms	0.74 ms	0.51 ms	0.74 ms	0.17 ms[2]
Convolution (3 x 3) Perform a general 3 x 3 convolution with arbitrary coefficients on an 8-bit source image and store results in an 8-bit destination image. Results are saturated.	1.8 ms	1.1 ms	0.63 ms	0.54 ms	0.15 ms[2]
Convolution (5 x 5) Same as above except with a 5 x 5 kernel.	4.6 ms	2.9 ms	1.6 ms	1.5 ms	0.21 ms[2]
Convolution (11 x 11) Same as above except with a 11 x 11 kernel.	19.8 ms	12.7 ms	7.1 ms	5.8 ms	0.96 ms[2]
Erosion/Dilation (3 x 3, predefined, binary) Perform a binary erosion/dilation on a 1-bit source image using a predefined 3 x 3 structuring element and store results in a 1-bit destination image.	0.08 ms	0.05 ms	0.04 ms	0.04 ms	0.09 ms[2]
Erosion/Dilation (3 x 3, predefined, grayscale) Same as above except perform a grayscale operation.	3.5 ms	0.17 ms	0.13 ms	0.16 ms	0.16 ms[2]
Erosion/Dilation (3 x 3, user-defined, binary) Perform a binary erosion/dilation on a 1-bit source image using an arbitrary 3 x 3 structuring element and store results in a 1-bit destination image.	0.34 ms	0.22 ms	0.15 ms	0.14 ms	0.10 ms[2]
Erosion/Dilation (3 x 3, user-defined, grayscale) Same as above except perform a grayscale erosion/dilation operation.	0.92 ms	0.51 ms	0.39 ms	0.51 ms	0.16 ms[2]
Erosion/Dilation (5 x 5, user-defined, binary) Perform a binary erosion/dilation on a 1-bit source image using an arbitrary 5 x 5 structuring element and store results in a 1-bit destination image.	1.4 ms	0.88 ms	0.58 ms	0.55 ms	0.09 ms[2]
Erosion/Dilation (5 x 5, user-defined, grayscale) Same as above except perform a grayscale erosion/dilation.	1.8 ms	1.1 ms	0.75 ms	1.1 ms	0.22 ms[2]
LUT map Perform a point-to-point LUT mapping operation for an 8-bit source image and store results in an 8-bit destination image.	0.59 ms	0.33 ms	0.54 ms	0.24 ms	0.28 ms[2]
Histogram Calculate the histogram of an 8-bit source image and store result in a 32-bit buffer.	0.58 ms	0.38 ms	0.28 ms	0.31 ms	0.31 ms
Lossy JPEG Compression (monochrome) Perform lossy JPEG compression on an 8-bit source image and store results in an 8-bit destination image.	2.5 ms	1.6 ms	1.2 ms	1.5 ms	1.5 ms

MIL/ActiveMIL Benchmarks (cont.)

Image Processing (cont.)	1.3 GHz Celeron™ M 512KB L2 Cache 400 MHz FSB PC2700 SDRAM (Matrox 4Sight M)	2.0 GHz Pentium™ M 2MB L2 Cache 400 MHz FSB PC2700 SDRAM (Matrox 4Sight M)	2.6 GHz Opteron™ 1 MB L2 Cache DDR1-400 SDRAM	3.6 GHz Xeon™ 1 MB L2 Cache 800 MHz FSB DDR2-400 SDRAM	Helios XA / XCL 133 MHz PA (with 3.6 GHz Xeon™)
Lossless JPEG Compression (monochrome) Perform lossless JPEG compression on an 8-bit source image and store results in an 8-bit destination image.	2.9 ms	1.9 ms	2.2 ms	2.3 ms	2.3 ms
Rotate (30°) Rotate by 30° an 8-bit source image and store results in 8-bit destination image.	1.2 ms	0.64 ms	0.90 ms	0.69 ms	0.69 ms
Warp Polynomial Warping using a first-order polynomial mapping with nearest neighbor interpolation on an 8-bit source image and store results in an 8-bit destination image.	1.2 ms	0.64 ms	0.90 ms	0.69 ms	0.69 ms
Geometric Model Finder [3, 4]					
Find a Model (1 model, 1 occurrence, very high speed, limited scaling) Find a single 128 x 128 model in an 8-bit image. The whole image is searched for a model rotated within 0-360° and scaled within 90-110% using the highest speed (lowest robustness and accuracy) setting.	6.2 ms	3.7 ms	2.8 ms	3.4 ms	3.4 ms
Find a Model (1 model, 1 occurrence, medium speed, limited scaling) Find a single 128 x 128 model in an 8-bit image. The whole image is searched for a model rotated within 0-360° and scaled within 90-110% using medium speed setting.	15.6 ms	10.2 ms	6.5 ms	8.1 ms	8.1 ms
Find a Model (1 model, 1 occurrence, medium speed, max. scaling) Find a single 128 x 128 model in an 8-bit image. The whole image is searched for a model rotated within 0-360° and scaled within 50-200% using medium speed setting.	16.2 ms	10.7 ms	7.1 ms	8.5 ms	8.5 ms
Find Models (1 model, 4 occurrences, medium speed, limited scaling) Same as above except find four occurrences of a single 128 x 128 model.	21.6 ms	13.5 ms	9.3 ms	11.8 ms	11.8 ms
Find Models (4 models, 4 occurrences, medium speed, limited scaling) Same as above except find a single occurrence of four 128 x 128 models.	25.6 ms	15.7 ms	11.1 ms	14.2 ms	14.2 ms
Pattern Matching (Normalized Grayscale Correlation) [3, 4]					
Find a Model (128 x 128, non-rotated) Find a 128 x 128 model in an 8-bit grayscale image. The whole image is searched for a model that is not rotated.	0.42 ms	0.19 ms	0.16 ms	0.19 ms	0.19 ms
Find a Model (128 x 128, -5° to +5°) Find a 128 x 128 model located at 0° in an 8-bit grayscale image. The whole image is searched for a a model rotated within +/-5°.	1.7 ms	0.89 ms	0.70 ms	0.89 ms	0.89 ms
Find a Model (32 x 32, non-rotated) As above except perform a pattern match of a 32 x 32 model.	2.1 ms	1.0 ms	0.71 ms	0.83 ms	0.83 ms
Find a Model (32 x 32, -5° to +5°) As above except perform a pattern match of a 32 x 32 model.	3.4 ms	1.8 ms	1.2 ms	1.4 ms	1.4 ms

MIL/ActiveMIL Benchmarks (cont.)

	1.3 GHz Celeron™ M 512KB L2 Cache 400 MHz FSB PC2700 SDRAM (Matrox 4Sight M)	2.0 GHz Pentium™ M 2MB L2 Cache 400 MHz FSB PC2700 SDRAM (Matrox 4Sight M)	2.6 GHz Opteron™ 1 MB L2 Cache DDR1-400 SDRAM	3.6 GHz Xeon™ 1 MB L2 Cache 800 MHz FSB DDR2-400 SDRAM	Helios XA / XCL 133 MHz PA (with 3.6 GHz Xeon™)
Edge Finder (4000 edge elements or edgels)[4]					
Extract contours	15.4 ms	9.0 ms	6.2 ms	8.0 ms	8.0 ms
Extract thin line crests	68.3 ms	24.3 ms	19.5 ms	24.8 ms	24.0 ms
Blob Analysis (100 blobs that occupy 25% of area)[4]					
Calculate Area	0.24 ms	0.15 ms	0.15 ms	0.12 ms	0.12 ms
Calculate Area and Binary Center of Gravity	0.28 ms	0.18 ms	0.17 ms	0.15 ms	0.15 ms
Calculate Area and Grayscale Center of Gravity	0.86 ms	0.56 ms	0.43 ms	0.45 ms	0.45 ms
Measurement					
Find an Edge Locate an edge in a 16 x 4 measurement region of an 8-bit image.	0.06 ms	0.02 ms	0.03 ms	0.07 ms	0.08 ms
Find Multiple Stripes Locate 24 stripes in a 128 x 16 measurement region of an 8-bit image.	0.17 ms	0.08 ms	0.08 ms	0.09 ms	0.15 ms
String Reader					
String Reading Read a 6 character string using a 28 character font within a 512 x 512 image region.	42.7 ms	26.4 ms	20.3 ms	25.0 ms	25.0 ms
OCR					
OCR Reading Read an unknown string of twelve 33 x 21 characters (no grammar rules) within a 404 x 54 image region.	10.6 ms	6.6 ms	4.9 ms	5.8 ms	5.8 ms
Verification Verify that a known string of 12 SEMI font characters (33 x 21) within a 404 x 54 image region can be read properly.	4.0 ms	0.83 ms	0.59 ms	0.84 ms	0.84 ms
Bar and Matrix Code Recognition					
Bar Code Reading Read a EAN13 bar code (no rotation).	0.28 ms	0.18 ms	0.15 ms	0.18 ms	0.27 ms
DataMatrix Reading Read a DataMatrix code.	3.7 ms	1.6 ms	2.1 ms	2.2 ms	2.3 ms

[1] Benchmarks for larger images do not necessarily scale linearly due to CPU cache effects.

[2] Performed using PA.

[3] Faster search speeds can be obtained by reducing accuracy.

[4] Search speeds will vary with image content.

MIL Command Listing and Description

This section provides an overview of each MIL module and a brief description of each MIL command. For a complete description of the syntax and use of each command, refer to the MIL Command Reference manual.

1D and 2D Code Reader module

Used to read (and write) various 1D and 2D code symbologies.

Commands	Command parameters	Description
McodeAlloc()	SystemId, CodeType, ControlFlag, CodeIdPtr	Allocate a code object.
McodeControl()	CodeId, ControlType, ControlValue	Control a code object.
McodeFree()	CodeId	Free a code object.
McodeGetResult()	CodeId, ResultType, ResultPtr	Get a result from a read or write operation.
McodeInquire()	CodeId, InquireType, UserVarPtr	Inquire about a code object parameter setting.
McodeRead()	CodeId, ImageBufId, ControlFlag	Read a specific type of code in an image.
McodeRestore()	FileName, SystemId, ControlFlag, CodeIdPtr	Restore a code object previously saved to a file.
McodeSave()	FileName, CodeId, ControlFlag	Save the specified code object in a file.
McodeStream()	MemPtrOrFileName, SystemId Operation, StreamType, Version ControlFlag, CodeIdPtr, SizeByteVarPtr	Load, restore, or save a code object from/to a file or memory.
McodeVerify()	CodeId, ImageBufId, String, ControlFlag	Compute the different quality-grades of the code in the specified source image.
McodeWrite()	CodeId, ImageBufId, String, ControlFlag	Encode an ASCII string.

1D and 2D code symbologies

For the McodeAlloc() command, the code type(s) that can be read or written include(s):

Code Type	Encoding Type	Error Correction
BC412	Standard encoding type	No error correction
Codabar	Standard encoding type	No error correction
Code39	ASCII encoding, Standard encoding type	No error correction; check-digit error correction
Code93	ASCII encoding	Check-digit error correction
Code128 (UCC/EAN128)	ASCII encoding	Check-digit error correction

Continued...

1D and 2D code symbologies (cont.)

For the McodeAlloc() command, the code type(s) that can be read or written include(s):

Code Type	Encoding Type	Error Correction
DataMatrix	Numeric encoding, Alpha encoding, AlphaNumericPunc encoding, AlphaNumeric encoding, ASCII encoding, IS08 encoding	10, 40, 50, 60, 70, 80,90, 100, 110, 120, 130, 140 or 200 error correction
EAN8	Numeric encoding	Check-digit error correction
EAN13	Numeric encoding	Check-digit error correction
Interleaved 2/5	Numeric encoding	No error correction; check-digit error correcion
Maxicode	Encoding mode 2, 3, 4, 5, 6	Reed Solomon error correction
MicroPDF417	Standard encoding type	Reed Solomon error correction
PDF417	Standard encoding type	Reed Solomon 1 - 8 error correction
Pharma	Numeric encoding	No error correction
Planet	Numeric encoding	Check-digit error correction
Postnet	Numeric encoding	Check-digit error correction
QR	QR code Model 1, 2 encoding	Lowest-level QR, Low-level QR, High-level QR, Highest-level QR
RSS	RSS 14, RSS 14 Stacked, RSS 14 Stacked Omni, RSS 14 Truncated, RSS Expanded RSS Expanded Stacked, RSS Limited encoding.	Check-digit error correction
UPC-A	Numeric encoding	Check-digit error correction
UPC-E	Numeric encoding	Check-digit error correction

Composite code symbologies

This code type is a composite of a 1D (RSS, UPC-A, UPC-E, EAN-8, EAN-13, or UCC/EAN128) and a 2D code type (PDF417 or MicroPDF417).

Application and System modules

Used to initialize and control the MIL application environment and system (frame grabber boards, vision processor boards, or host system) respectively. The Application module includes control of integrated debugging features, system resource compensation, command threads and related events, as well as a timer function.

Commands	Command parameters	Description
MappAlloc()	InitFlag, ApplicationIdPtr	Allocate a MIL application.
MappAllocDefault()	InitFlag, ApplicationIdPtr, SystemIdPtr, DisplayIdPtr, DigIdPtr, ImageBufIdPtr	Allocate MIL application defaults.
MappControl()	ControlType, ControlFlag	Control an application environment setting.
MappFree()	ApplicationId	Free a MIL application.
MappFreeDefault()	ApplicationId, SystemId, DisplayId, DigId, ImageBufId	Free MIL application defaults.
MappGetError()	ErrorType, ErrorPtr	Get error codes and related information.
MappGetHookInfo()	EventId, InfoType, UserVarPtr	Get information about a hooked event.
MappHookFunction()	HookType, HookHandlerPtr, ExpansionFlag	Hook a function to an event.
MappInquire()	InquireType, UserVarPtr	Inquire about the application parameter setting.
MappTimer()	ControlValue, TimePtr	Control the MIL timer.
MsysAlloc()	SystemTypePtr, SystemNum, InitFlag, SystemIdPtr	Allocate a hardware system.
MsysControl()	SystemId, ControlType, ControlFlag	Control system behavior.
MsysFree()	SystemId	Free a system.
MsysHookFuntion()	SystemId, HookType, HookHandlerPtr, UserDataPtr	Hook a function to a system event.
MsysInquire()	SystemId, ParamToInquire, UserVarPtr	Inquire about a system parameter setting.

Blob analysis module

Used to identify and measure connected components (blobs) in an image.

Commands	Command parameters	Description
MblobAllocFeatureList()	SystemId, FeatureListIdPtr	Allocate a blob analysis feature list.
MblobAllocResult()	SystemId, BlobResIdPtr	Allocate a blob analysis result buffer.
MblobCalculate()	BlobIdentImageId, GrayImageId, FeatureListId, BlobResId	Perform blob analysis calculations.
MblobControl()	BlobResId, Procmode, Value	Control a blob analysis processing mode setting.
MblobDraw()	GraphContId, ResultId, DestImageId, Operation, Label, ControlFlag	Draw features of specified blob results in an image buffer.
MblobFill()	BlobResId, DestImageBufId, Criteria, Value	Draw blobs that meet a specified fill criterion.
MblobFree()	BlobId	Free the blob analysis result buffer or the feature list.
MblobGetLabel()	BlobResId, XPos, YPos, LabelVarPtr	Get the label value of a blob at a specific position.
MblobGetNumber()	BlobResId, CountVarPtr	Get the number of currently included blobs.
MblobGetResult()	BlobResId, Feature, TargetArrayPtr	Read feature values of the included blobs.
MblobGetResultSingle()	BlobResId, LabelVal, Feature, TargetArrayPtr	Read the feature value of a single blob.
MblobGetRuns()	BlobResId, LabelVal, ArrayType, RunXPtr, RunYPtr, RunLengthPtr	Get the blob run-length encoding information.
MblobInquire()	BlobResId, InquireType, UserVarPtr	Inquire about a blob analysis processing mode.
MblobLabel()	BlobResId, DestImageBufId, Mode	Draw a labeled image.
MblobReconstruct()	SrcImageBufId, SeedImageBufId, DestImageBufId, Operation, ProcMode	Reconstruct blobs (or blob holes) in an image buffer.
MblobSelect()	BlobResId, Operation, Feature, Condition, CondLow, CondHigh	Select blobs for calculations and result retrieval.
MblobSelectFeature()	FeatureListId, Feature	Select feature(s) to be calculated. See complete feature list on the following page.
MblobSelectFeret()	FeatureListId, Angle	Add Feret angle to the feature list.
MblobSelectMoment()	FeatureListId, MomType, XMomOrder, YMomOrder	Add specified moment calculations to the feature list.

Blob features

For the MblobSelectFeature() command, the feature(s) that can be calculated include(s):

M_AREA, the number of foreground pixels in a blob.
M_BOX_X_MIN, M_BOX_Y_MIN, M_BOX_X_MAX, M_BOX_Y_MAX, the coordinates of the extreme left, top, right and bottom pixels, respectively, of a blob.
M_BREADTH, a measure of the true breadth of an object.
M_CHAIN_INDEX, this is the index which differentiates chains in a blob.
M_CHAIN_Y, M_CHAIN_X, these are the x and y coordinates of each chained pixel.
M_COMPACTNESS, a minimum for a circle (1.0) and is derived from the perimeter and area.
M_CONVEX_PERIMETER, an approximation of the perimeter of the convex hull of a blob.
M_ELONGATION, equal to M_LENGTH over M_BREADTH.
M_EULER_NUMBER, the number of blobs minus the number of holes.
M_FERET_X, M_FERET_Y, the dimensions of the minimum bounding box of a blob in the horizontal and vertical directions (respectively).
M_FERET_MIN_DIAMETER, the smallest Feret diameter found after checking a certain number of angles.
M_FERET_MIN_ANGLE, the angle at which the minimum Feret diameter is found.
M_FERET_MAX_DIAMETER, the largest Feret diameter found after checking a certain number of angles.
M_FERET_MAX_ANGLE, the angle at which the maximum Feret diameter is found.
M_FERET_MEAN_DIAMETER, the average Feret diameter at all the angles checked.
M_FERET_ELONGATION, a measure of the shape of a blob.
M_FIRST_POINT_X, M_FIRST_POINT_Y, a unique point for each object, which is always on the perimeter of the object.
M_INTERCEPT _0, _45, _90, _135, the number of times that a transition from background to foreground occurs at the given angle for the entire blob.
M_LABEL_VALUE, the label value for each blob in an image.
M_LENGTH, a measure of the true length of an object.
M_NUMBER_OF_CHAINED_PIXELS, this is the number of chained pixels for all blobs or a specified blob.
M_NUMBER_OF_HOLES, the number of holes in a blob.
M_NUMBER_OF_RUNS, the total number of horizontal strings of consecutive foreground pixels in a blob.
M_PERIMETER, the total length of edges in a blob (including the edges of any holes).
M_ROUGHNESS, a measure of how rough a blob is.
M_X_MIN_AT_Y_MIN, M_X_MAX_AT_Y_MAX, M_Y_MIN_AT_X_MAX, M_Y_MAX_AT_X_MIN, these values, together with the four box coordinates, give four contact points on the convex perimeter of the object.

For a grayscale image:
M_MEAN_PIXEL, the mean pixel value in a blob.
M_MIN_PIXEL, the minimum pixel value found in a blob.
M_MAX_PIXEL, the maximum pixel value found in a blob.
M_SIGMA_PIXEL, the standard deviation of pixel values in a blob.
M_SUM_PIXEL, the sum of all pixel values in a blob.
M_SUM_PIXEL_SQUARED, the sum of the squares of each pixel value in a blob.

The following features have two different definitions: a binary one, where all pixels are considered equal; and a grayscale one, where pixels are weighted by their value in the gray scale image.
M_CENTER_OF_GRAVITY_X, the x position of the center of gravity of a blob.
M_CENTER_OF_GRAVITY_Y, the y position of the center of gravity of a blob.
M_MOMENT_Xn_Ym and M_MOMENT_CENTRAL_Xn_Ym for central moments; coordinates are relative to each blob's center of gravity; ordinary moments use coordinates relative to the image origin.
M_AXIS_PRINCIPAL_ANGLE, the angle at which a blob has the least moment of inertia.
M_AXIS_SECONDARY_ANGLE, the angle perpendicular to M_AXIS_PRINCIPAL_ANGLE.

The following predefined values let the user select groups of features in a single call:
M_BOX, adds all 4 box features plus x and y Ferets.
M_CONTACT_POINTS, adds first point and other contact features.
M_CENTER_OF_GRAVITY, adds both x and y coordinates of the center of gravity.
M_ALL_FEATURES, adds all features (except general Feret and general moment).
M_NO_FEATURES, removes all features (except label value).
M_CHAINS, adds all 4 chain features.

You can add the following sorting options to a feature to specify it as a sorting key during result retrieval:
M_SORTn_DOWN, specifies the feature as the nth sorting key (in a descending order) where n is an integer between 1 and 3.
M_SORTn_UP, specifies the feature as the nth sorting key (in an ascending order) where n is an integer between 1 and 3.
M_NO_SORT, removes the specified sorting key.

Buffer and Data generation modules

Used to allocate and control a data buffer, and to generate data for the LUT and the warp function. The Buffer module includes control of a child buffer (ROI), buffer compression and decompression, custom kernel or structuring element, and buffer archiving and retrieving.

Commands	Command parameters	Description
MbufAlloc1d()	SystemId, SizeX, Type, Attribute, BufIdPtr	Allocate a 1D data buffer.
MbufAlloc2d()	SystemId, SizeX, SizeY, Type, Attribute, BufIdPtr	Allocate a 2D data buffer.
MbufAllocColor()	SystemId, SizeBand, SizeX, SizeY, Type, Attribute, BufIdPtr	Allocate a color data buffer.
MbufBayer()	SrcImageBufId, DestImageBufId, WhiteBalanceCoefficientsID, ControlFlag	Decode the color information of a single-band, Bayer color-encoded image.
MbufChild1d()	ParentBufId, OffX, SizeX, BufIdPtr	Allocate a 1D child data buffer.
MbufChild2d()	ParentBufId, OffX, OffY, SizeX, SizeY, BufIdPtr	Allocate a child buffer from a specific region of the parent buffer.
MbufChildColor()	ParentBufId, Band, BufIdPtr	Allocate a color-band child data buffer within a color parent buffer.
MbufChildColor2d()	ParentBufId, Band, OffX, OffY, SizeX, SizeY, BufIdPtr	Allocate a child data buffer within a color parent buffer.
MbufChildMove()	BufferID, OffsetX, OffsetY,SizeX, SizeY ControlFlag	Move and resize a child buffer within the parent buffer
MbufClear()	DestImageBufId, Color	Clears a buffer to a specified color.
MbufControl()	BufId, ControlType, ControlValue	Control specified buffer features.
MbufControlNeighborhood()	BufId, OperationFlag, OperationValue	Change the value of an operation flag associated with a custom kernel or structuring element.
MbufControlRegion()	BufId, OffsetX, OffsetY, SizeX, SizeY, Band, ControlType, ControlValue,	Control a specified region of a buffer.
MbufCopy()	SrcBufId, DestBufId	Copy data from one buffer to another (optionally with compression or format conversion).
MbufCopyClip()	SrcBufId, DestBufId, DestOffX, DestOffY	Copy buffer, clipping data outside destination buffer.
MbufCopyColor()	SrcBufId, DestBufId, Band	Copy one or all bands of an image buffer.
MbufCopyColor2d()	SrcBufId, DestBufId, ScrBand, ScrOffX, ScrOffY, DstBand, DstOffX, DstOffY, SizeX, SizeY	Copy a 2D region of one or all bands of an image buffer to another buffer.
MbufCopyCond()	SrcBufId, DestBufId, CondBufId, Condition, CondValue	Copy conditionally the source buffer to the destination buffer.
MbufCopyMask()	SrcBufId, DestBufId, MaskValue	Copy buffer with mask.
MbufCreateColor()	SystemId, SizeBand, SizeX, SizeY, Type, ControlFlag, Pitch, ArrayOfDataPtr BufIdPtr	Create a color data buffer.
MbufCreate2d()	SystemId, SizeX, SizeY, Type, Attribute, ControlFlag, Pitch, DataPtr, BufIdPtr	Create a two-dimensional data buffer.
MbufDiskInquire()	FileName, ParamToInquire, UserVarPtr	Inquire about the buffer data in a file.
MbufExport()	FileName, FileFormat, SrcBufId	Export a data buffer to a file.
MbufExportSequence()	FileName, FileFormatId, BufArrayPtr, NumberOfImages,FrameRate, ControlFlag	Export a sequence of image buffers to an AVI file.
MbufFree()	BufId	Free a data buffer.
MbufGet1d()	SrcBufId, OffX, SizeX, UserArrayPtr	Get data from a 1D area of a buffer and place it in a user-supplied array.

Buffer and Data generation modules (continued)

Commands	Command parameters	Description
MbufGet2d()	SrcBufId, OffX, OffY, SizeX, SizeY, UserArrayPtr	Get data from a 2D area of a buffer and place it in a user-supplied array.
MbufGet()	SrcBufId, UserArrayPtr	Get data from a buffer and place it in a user-supplied array.
MbufGetArc()	ImageBufId, XCenter, YCenter, XRad YRad, StartAngle, EndAngle, NbPixelsPtr, UserArrayPtr	Read the pixels along a specified arc and store their values in a user-defined array.
MbufGetColor()	SrcBufId, DataFormat, Band, UserArrayPtr	Get data from one or all bands of a buffer and place it in a user-supplied array.
MbufGetColor2d()	SrcBufId, DataFormat, Band, OffX, OffY, SizeX, SizeY, UserArrayPtr	Get data from a region of one or all bands of a buffer and place it in a user-supplied array.
MbufGetHookInfo()	EventId, InfoType, UserVarPtr	Get information about a hook event.
MbufHookFunction()	BufferId, HookType, HookHandlerPtr, UserDataPtr	Hook a function to a buffer event.
MbufGetLine()	ImageBufId, StartX, StartY, EndX, EndY, Mode, NumPixelsPtr, UserArrayPtr	Read the pixels of a theoretical line between specified coordinates, count them, and store them in a user-defined array.
MbufImport()	FileName, FileFormat, Operation, SystemId, BufIdPtr	Import data from a file into a data buffer.
MbufImportSequence()	FileName, FileFormatId, Operation, SystemId, BufArrayPtr, StartImage, NumberOfImages, ControlFlag	Import a sequence of images from an AVI file into separate image buffers.
MbufInquire()	BufId, ParamToInquire, UserVarPtr	Inquire about a data buffer parameter setting.
MbufLoad()	FileName, BufId	Load data from a file into a data buffer.
MbufPut()	DestBufId, UserArrayPtr	Put data from a user-supplied array into a data buffer.
MbufPutColor()	DestBufId, DataFormat, Band, UserArrayPtr	Put data from a user-supplied array into one or all bands of a data buffer.
MbufPutColor2d()	DestBufId, DataFormat, Band, OffX, OffY, SizeX, SizeY, UserArrayPtr	Put data from a user-supplied array into a region of one or all bands of a data buffer.
MbufPutLine()	ImageBufId, StartX, StartY, EndX, EndY, Mode, NumbPixelsPtr, UserArrayPtr	Write a specified series of pixels within specified coordinates, along a theoretical line.
MbufPut1d()	DestBufId, OffX, SizeX, UserArrayPtr	Put data from a user-supplied array into a 1D area of a buffer.
MbufPut2d()	DestBufId, OffX, OffY, SizeX, SizeY, UserArrayPtr	Put data from a user-supplied array into a 2D area of a buffer.
MbufRestore()	FileName, SystemId, BufIdPtr	Restore data from a file into an automatically allocated data buffer.
MbufSave()	FileName, BufId	Save a data buffer in a file, using the MIL output file format.
MbufTransfer()	SrcBufId, DestBufId, SrcOffX, SrcOffY SrcBand, DestOffX, DestOffY, DestSizeX DestSizeY DestBand, TransferFunction TransferType, OperationFlag ExtraParameter	Copy a 2D region of one or all bands from the source buffer into a 2D region of one or all bands in the destination buffer, using a specified transfer function and transfer type file format.
MgenLutFunction()	LutBufId, Func, a, b, c, StartIndex, StartXValue, EndIndex	Generate data into a LUT buffer using a specified standard mathematical function.
MgenLutRamp()	LutId, StartIndex, StartValue, EndIndex, EndValue	Generate ramp data into a LUT buffer.
MgenWrapParameters()	InWarpParameter, OutXLutOrCoef, OutYLut, OperationMode, Transform, Val1, Val2	Generate coefficients or LUTs for use with MimWarp().

Calibration module

Used to convert coordinates or measurements from pixel to real-world units, as well as to correct distortions in an image.

Commands	Command parameters	Description
McalAlloc()	Mode, ModeFlag, CalibrationIdPtr	Allocate a calibration object.
McalAssociate() to/from	CalibrationId, ImageOrDigitizerId,	Associate/disassociate a calibration object ControlFlag an image or digitizer.
McalControl()	CalibrationId, ControlType, ControlValue	Control a calibration object parameter setting.
McalFree()	CalibrationId	Free a calibration object.
McalGrid()	CalibrationId, SrcImageBufId, GridOffsetX, GridOffsetY, GridOffsetZ, RowNumber, ColumnNumber, RowSpacing, ColumnSpacing, Mode, ModeFlag	Calibrate your imaging setup using a grid.
McalInquire()	CalibrationOrMilId, InquireType, UserVarPtr	Inquire about a calibration object setting or about the calibration object associated to an image or digitizer.
McalList()	CalibrationId, XPixArray, YPixArray, XWorldArray, YWorldArray, ZWorld, NumPoint, Mode, ModeFlag	Calibrate your imaging setup using a list of coordinates.
McalRelativeOrigin()	CalibrationId, XOffset, YOffset, ZOffset, AngularOffset, ControlFlag	Change the origin and/or orientation of a relative coordinate system.
McalRestore()	FileName, ControlFlag, CalibrationIdPtr	Restore a calibration object from a file.
McalSave()	FileName, CalibrationId, ControlFlag	Save a calibration object to a file.
McalStream()	MemPtrOrFileName, SystemId, Operation, StreamType, Version, ControlFlag, CalibrationIdPtr SizeByteVarPtr	Load, restore, or save a calibration object from/to a file or a memory.
McalTransformCoordinate()	CalibrationOrMilId, TransformType, X, Y, ResXPtr, ResYPtr	Convert coordinates between world and pixel values.
McalTransformCoordinateList()	CalibrationOrMilId, TransfromType, NumPoints, SrcXPtr, SrcYPtr, ResXPtr, ResYPtr	Convert a list of coordinates between their world and pixel values.
McalTransformImage()	SrcImageBufId, DestImageBufId, CalibrationId, InterpolationMode, OperationType, ControlFlag	Physically transform an image to remove any distortions.
McalTransformResult()	CalibrationOrMilId, TransformType, ResultType, Result, ResResult	Convert a result between world and pixel value.

Digitizer module

Used to initialize and control a digitizer (image capture device). This module includes control of capture mode (trigger, frame/field, blocking/non-blocking), image scaling and cropping, input channel, input LUT, analog settings (references, hue, saturation, and brightness) as well as events for callback functions.

Commands	Command parameters	Description
MdigAlloc()	SystemId, DigNum, DataFormat, InitFlag, DigIdPtr	Allocate a digitizer.
MdigChannel()	DigId, Channel	Select the active input channel of a digitizer.
MdigControl()	DigId, ControlType, Value	Control the specified digitizer.
MdigFocus()	DigId, DestImageBufId, FocusImageRegionBufId, FocusHookPtr, UserDataPtr, MinPosition, StartPosition, MaxPosition, MaxPositionVariation, ProcMode, ResultPtr	Adjust a camera's lens motor to a position which provides optimum focus.
MdigFree()	DigId	Free a digitizer.
MdigGrab()	ScrDigId, DestImageBufId	Grab data from an input device into a buffer.
MdigGrabContinuous()	DigId, DestImageBufId	Grab data continuously from an input device.
MdigGrabWait()	DigId, Flag	Wait for the end of the grab in progress.
MdigHalt()	DigId	Halt a continuous grab from an input device.
MdigHookFunction()	DigId, HookType, HookHandlerPtr, UserDataPtr	Hook a function to a digitizer event.
MdigInquire()	DigId, InquireType, UserVarPtr	Inquire about a digitizer parameter setting.
MdigLut()	DigId, LutBufId	Copy a LUT buffer to a digitizer LUT.
MdigProcess()	DigId, DestImageArrayPtr, ImageCount Operation, OperationFlag, HookHandlerPtr, UserDataPtr	Grabs a sequence of images and process them with a user-defined function as they are grabbed.
MdigReference()	DigId, ReferenceType, ReferenceLevel	Select digitization reference level.

Display module

Used to initialize and control an image display. This module includes control of image display windows, graphics overlay, output LUT, image pan, scroll, and zoom.

Commands	Command parameters	Description
MdispAlloc()	SystemId, DispNum, DispFormat, InitFlag, DisplayIdPtr	Allocate a display.
MdispControl()	DisplayId, ControlType, ControlValue	Control the MIL display.
MdispFree()	DisplayId	Free a display.
MdispHookFunction()	DisplayId, HookType, HookHandlerPtr, UserDataPtr	Hook a function to a display event.
MdispInquire()	DisplayId, InquireType, UserVarPtr	Inquire about a display parameter setting.
MdispLut()	DisplayId, LutBufId	Copy a LUT buffer to a display output LUT.
MdispPan()	DisplayId, XOffset, YOffset	Pan and scroll a display.
MdispSelect()	DisplayId, ImageBufId	Select an image buffer to display.
MdispSelectWindow()	DisplayId, ImageBufId, ClientWindowHandle	Select an image buffer to display in a user-defined window.
MdispZoom()	DisplayId, XFactor, YFactor	Zoom a display.

Edge Finder module

Used to extract and analyze object contours or thin curvilinear features.

Commands	Command parameters	Description
MedgeAlloc()	SystemId, EdgeFinderType, ControlFlag, ContextIdPtr	Allocate an Edge Finder context.
MedgeAllocResult()	SystemId, ControlFlag, EdgeResultIdPtr,	Allocate an Edge Finder result buffer.
MedgeCalculate()	ContextId, SourceImageId, SourceDeriv1Id, SourceDeriv2Id, SourceDeriv3Id, EdgeResultId, ControlFlag	Perform edge extraction and feature calculations.
MedgeControl()	ContextOrResultId, ControlType, ControlValue	Control an Edge Finder context or an Edge Finder result buffer.
MedgeDraw()	GraphContId, EdgeResultId, DestImageId, Operation, IndexOrLabel, ControlFlag	Draw specific edge features in the destination image buffer.
MedgeFree()	ObjectId	Free an Edge Finder context or an Edge Finder result buffer.
MedgeGetNeighbors()	EdgeResultId, SizeOfArray, SrcArrayXPtr SrcArrayYPtr, SrcArrayAnglePtr, DstArrayXPtr, DstArrayYPtr, DstArrayIndexPtr, DstArrayLabelPtr, ControlFlag	Get edgels from an Edge Finder result buffer that are the closest neighbors to a list of user-specified point coordinates.
MedgeGetResult()	EdgeResultId, EdgeIndexOrLabelValue, ResultType, FirstResultArrayPtr, SecondResultArrayPtr	Get results of the included edges from an Edge Finder result buffer.
MedgeInquire()	ContextOrResultId, InquireType, UserVarPtr	Inquire about an Edge Finder context or an Edge Finder result buffer.
MedgeMask()	ContextId, MaskImageId, ControlFlag	Mask regions of the image.
MedgeRestore()	Filename, SystemId, ControlFlag, ContextIdPtr	Restore an Edge Finder context from disk.
MedgeSave()	FileName, ContextOrResultId, ControlFlag	Save an Edge Finder context to a file, or save edge chains and/or edge approximations from an Edge Finder result buffer to a CAD (Computer-Aided Design) file.
MedgeSelect()	EdgeResultId, Operation, Feature, Condition, Param1, Param2	Select edges for calculations and result retrieval.
MedgeStream()	MemPtrOrFileName, SystemId, Operation, StreamType, Version, ControlFlag, ContextOrResultIdPtr, SizeByteVarPtr	Load, restore, or save an Edge Finder context from/to a file or memory, or save calculated edges from an Edge Finder result buffer to a file or memory in DXF format.

Edge features

For the MedgeGetResults() command, the feature(s) that can be calculated include(s):

M_AVERAGE_STRENGTH, returns the average strength value of each edge.

M_BOX_X_MAX, returns the X-coordinate of each edge's right-most edgel.

M_BOX_X_MIN, returns the X-coordinate of each edge's left-most edgel.

M_BOX_Y_MAX, returns the Y-coordinate of each edge's bottom-most edgel.

M_BOX_Y_MIN, returns the Y-coordinate of each edge's top-most edgel.

M_BULGES, returns the bulge values between vertices.

M_CENTER_OF_GRAVITY, returns the coordinates of each edge's center of gravity.

M_CENTER_OF_GRAVITY_X, returns the X-coordinate of each edge's center of gravity.

M_CENTER_OF_GRAVITY_Y, returns the Y-coordinate of each edge's center of gravity.

M_CIRCLE_FIT_CENTER_X, returns the X-coordinate of the center of the circle that is the best fit for each edge.

M_CIRCLE_FIT_CENTER_Y, returns the Y-coordinate of the center of the circle that is the best fit for each edge.

Continued...

Edge features (continued)

M_CIRCLE_FIT_COVERAGE, returns the coverage of the circle that is the best fit for each edge.
M_CIRCLE_FIT_ERROR, returns the fit error of the circle that is the best fit for each edge.
M_CIRCLE_FIT_RADIUS, returns the radius of the circle that is the best fit for each edge.
M_CHAIN, returns the coordinates of the edge(s)'s edgels.
M_CHAIN_ANGLE, returns the direction of the edge(s)'s edgels.
M_CHAIN_CODE, returns the edge(s)'s chain code.
M_CHAIN_INDEX, returns the index of the edge(s)'s edgels.
M_CHAIN_MAGNITUDE + M_CHAIN_ANGLE, returns the magnitude values and the angle values of the edge(s)'s edgels.
M_CHAIN_MAGNITUDE, returns the magnitude values of the edge(s)'s edgels.
M_CHAIN_X, Y, returns the X or Y-coordinates of the edge(s)'s edgels.
M_CLOSURE, Returns the closure of each edge.
M_CONVEX_PERIMETER, returns the convex elongation of each edge.
M_ELLIPSE_FIT_ANGLE , returns the angle of the ellipse that is the best fit for each edge.
M_ELLIPSE_FIT_CENTER_X, Y, returns the X or Y-coordinate of the center of the ellipse that is the best fit for each edge.
M_ELLIPSE_FIT_COVERAGE, returns the coverage of the ellipse that is the best fit for each edge.
M_ELLIPSE_FIT_ERROR, returns the fit error of the ellipse that is the best fit for each edge.
M_ELLIPSE_MAJOR_AXIS, returns the major axis of the ellipse that is the best fit for each edge.
M_ELLIPSE_MINOR_AXIS, returns the minor axis of the ellipse that is the best fit for each edge.
M_FAST_LENGTH, returns the fast length of each edge.
M_FERET_BOX, returns the X- and Y-Feret values of each edge.
M_FERET_ELONGATION, returns the Feret elongation of each edge.
M_FERET_MAX_ANGLE, returns the maximum Feret angle of each chain, in degrees.
M_FERET_MAX_DIAMETER, returns the maximum Feret diameter of each edge.
M_FERET_MEAN_DIAMETER, returns the average Feret diameter at all the angles checked.
M_FERET_MIN_ANGLE, returns the minimum Feret angle of each chain.
M_FERET_MIN_DIAMETER, returns the minimum Feret diameter of each edge.
M_FERET_X, Y, returns the X or Y-Feret value of each edge.
M_FIRST_POINT, returns the coordinates of each edge's first point.
M_FIRST_POINT_X, Y, returns the X or Y-coordinate of each edge's first point.
M_GENERAL_FERET, returns the general Feret of each edge.
M_LABEL_VALUE, returns the label value of each edge in an image.
M_LENGTH, returns the length of each edge.
M_LINE_FIT_A , _B, _C, returns the A, B or C variable of the line that is the best fit for each edge.
M_LINE_FIT_ERROR, returns the fit error of the line that is the best fit for each edge.
M_MOMENT_ELONGATION, returns the moment elongation of each edge.
M_MOMENT_ELONGATION_ANGLE, returns the angle of the principle axis along each edge's moment elongation.
M_NUMBER_OF_CHAINED_EDGELS, returns the total number of edgels in the edge(s).
M_NUMBER_OF_CHAINS, returns the number of included edges.
M_NUMBER_OF_VERTICES, returns the total number of chain approximation vertices in the edge(s).
M_POSITION, returns the X- and Y-position of each edge.
M_POSITION_X, Y, returns the X or Y-position of each edge.
M_SIZE, returns the number of edgels in each edge.
M_STRENGTH, returns the strength value of each edge.
M_TORTUOSITY, returns the tortuosity measure of each edge.
M_VERTICES, returns the coordinates of the chain approximation's vertices.
M_VERTICES_X, Y, returns the X or Y-coordinates of the chain approximation's vertices.
M_X_MAX_AT_Y_MAX, returns the maximum X-coordinate at the maximum Y-coordinate of each edge.
M_X_MIN_AT_Y_MIN, returns the minimum X-coordinate at the minimum Y-coordinate of each edge.
M_Y_MAX_AT_X_MIN, returns the maximum Y-coordinate at the minimum X-coordinate of each edge.
M_Y_MIN_AT_X_MAX, returns the minimum Y-coordinate at the maximum X-coordinate of each edge.

Function Developer’s Toolkit

The MIL Function Developer's Toolkit allows programmers to define functions to extend MIL's functionality. Using this toolkit, you can implement functions and integrate them directly into the MIL library, where they behave like standard MIL functions (e.g., respecting error handling and tracing).

Commands	Command parameters	Description
MfuncAlloc()	FunctionName, ParameterNumber, SlaveFunctionPtr, Reserved1, Reserved2, SlaveFunctionOpcode, InitFlag, FuncIdPtr	Allocate a MIL function context for your user-defined function.
MfuncAllocId()	FunctionId, ObjectType, ObjectPtr	Associate a MIL identifier with a user-defined object.
MfuncCall()	FunctionId	Execute the slave function.
MfuncErrorReport()	FunctionId, ErrorCode, ErrorMessage, ErrorSubMessage1, ErrorSubMessage2, ErrorSubMessage3	Report an error message.
MfuncFree()	FunctionId	Free a MIL function context.
MfuncFreeId()	FunctionId, ObjectId	Free the MIL identifier associated with a user-defined MIL object.
MfuncInquire()	ObjectId, InquireType, UserVarPtr	Retrieve information on a user-defined MIL object.
MfuncParamCheck()	FunctionId	Verify whether parameter checking is required.
MfuncParamDouble()	FunctionId, ParamIndex, ParamValue	Register a parameter of type double.
MfuncParamId()	FunctionId, ParamIndex, ParamValue ParamIs, RequiredAttribute	Register a MIL_ID parameter.
MfuncParamIdPointer()	FunctionId, ParamIndex, ParamValue, ParamIs, ParamAttribute	Register a MIL_ID pointer parameter
MfuncParamLong()	FunctionId, ParamIndex, ParamValue	Register a parameter of type long.
MfuncParamPointer()	FunctionId, ParamIndex, ParamValue Size, Attribute	Register a pointer parameter.
MfuncParamString()	FunctionId, ParamIndex, ParamValue Size, Attribute	Register a null-terminated string parameter.
MfuncParamValue()	FunctionId, ParamIndex, ParamValuePtr	Read the value of the specified MIL function parameter.

Geometric Model Finder module

Use geometric features (i.e., contours) to find models in an image. This module includes functions to define models, control search strategy, and save and restore models.

Commands	Command parameters	Description
MmodAlloc()	SystemId, ModelFinderType, ControlFlag, ContextIdPtr	Allocate a model finder context.
MmodAllocResult()	SystemId, ControlFlag, ModResultIdPtr	Allocate a model finder result buffer.
MmodControl()	ContextId, Index, ControlType, ControlValue	Control a model finder context setting.
MmodDefine()	ContextId, ModelType, Param1, Param2, Param3, Param4, Param5	Add a model to, or delete model from, a model finder context.
MmodDefineFromFile()	ContextId, FileType, Filename, ControlFlag	Defines a model from a file and adds it to a Model Finder context.
MmodDraw()	GraphContId, ContextOrResultId, DestImageId, Operation, Index, ControlFlag	Draw features of specific models or result occurrences in an image buffer.
MmodFind()	ContextId, TargetImageId, ModResultId	Search for the model(s) of the specified Model Finder context in a target image buffer or in an Edge Finder result buffer.
MmodFree()	ObjectId	Free a measurement context, marker, or result buffer.
MmodGetResult()	ResultId, ResultIndex, ResultType, ResultArrayPtr	Get the model finder result values.
MmodInquire()	ImageBufId, ModelId, FindResultId, ResultRange	Inquire information from a specified model finder context.
MmodMask()	ContextId, Index, MaskBufferId, MaskType, ControlFlag	Mask regions of a model result buffer.
MmodPreprocess()	ContextId, ControlFlag	Preprocess a model finder context.
MmodRestore()	FileName, SystemId, ControlFlag, ContextIdPtr	Restore a model finder context from disk.
MmodSave()	FileName, ContextId, ControlFlag	Save a model finder context to a file.
MmodStream()	MemPtrOrFileName, SystemId, Operation, StreamType, Version ControlFlag, ContextIdPtr, SizeByteVarPtr	Load, restore, or save a Model Finder context from/to a file or a memory.

Graphics module

Used to create drawings and text annotations in an image. This module provides a set of graphics primitives (arc, circle, line, and rectangle), control of color (foreground, background, fill), and text (font, color, size).

Commands	Command parameters	Description
MgraAlloc()	SystemId, GraphContIdPtr	Allocate a graphics context.
MgraArc()	GraphContId, DestImageBufId, XCenter, YCenter, XRad, YRad, StartAngle, EndAngle	Draw an arc.
MgraArcFill()	GraphContId, DestImageBufId, XCenter, YCenter, XRad, YRad, StartAngle, EndAngle	Draw a filled elliptic arc.
MgraBackColor()	GraphContId, BackgroundColor	Sets the background color of a graphics context.
MgraClear()	GraphContId, DestImageBufId	Clear an image buffer to a specified foreground color.
MgraColor()	GraphContId, ForegroundColor	Sets the foreground color of a graphics context.
MgraControl()	GraphContId, ControlType, Control	Control the specified graphics context.
MgraDot()	GraphContId, DestImageBufId, XPos, YPos	Draw a dot.
MgraDots()	GraphContId, DestImageBufId, NumberOfDots, XPosArray, YPosArray, ControlFlag	Draw one or more dots
MgraFill()	GraphContId, DestImageBufId, XStart, YStart	Perform a boundary-type seed fill.
MgraFont()	GraphContId, FontName	Associate a text font with a graphics context.
MgraFontScale()	GraphContId, XFontScale, YFontScale	Set the font scale of a graphics context.
MgraFree()	GraphContId	Free a graphics context.
MgraInquire()	GraphContId, InquireType, UserVarPtr	Inquire about the graphics parameters.
MgraLine()	GraphContId, DestImageBufId, XStart, YStart, XEnd, YEnd	Draw a line.
MgraLines()	GraphContId, DestImageBufId, NumberOfLines, XStartArray, YStartArray, XEndArray, YEndArray ControlFlag	Draw one or more lines.
MgraRect()	GraphContId, DestImageBufId, XStart, YStart, XEnd, YEnd	Draw a rectangle.
MgraRectFill()	GraphContId, DestImageBufId, XStart, YStart, XEnd, YEnd	Draw a filled rectangle.
MgraText()	GraphContId, DestImageBufId, XStart, YStart, String	Write text.

Image processing module

Used to perform filtering, morphological, point-to-point, segmentation, and statistical operations on an image. This module also includes geometric, color space, and domain transforms, as well as other image processing primitives.

Commands	Command parameters	Description
MimAllocResult()	SystemId, NbEntries, ResultType, ImResultIdPtr	Allocate an image processing result buffer.
MimArith()	Src1ImageBufId, Src2ImageBufId, DestImageBufId, Operation	Perform a point-to-point arithmetic operation.
MimArithMultiple()	Src1ImageBufId, Src2ImageBufId, Src3ImageBufId, Src4ImageBufId, Src5ImageBufId, DestImageBufId, Operation, OperationFlag	Perform a point-to-point arithmetic operation using multiple source images.
MimBinarize()	SrcImageBufId, DestImageBufId, Condition, CondLow, CondHigh	Perform a point-to-point binary thresholding operation.
MimClip()	SrcImageBufId, DestImageBufId, Condition, CondLow, CondHigh, WriteLow, WriteHigh	Perform a point-to-point clipping operation.
MimClose()	SrcImageBufId, DestImageBufId, NbIteration, ProcMode	Perform a binary or grayscale closing-type morphological operation.
MimConnectMap()	SrcImageBufId, DestImageBufId, LutBufId	Perform a 3 by 3 binary connectivity mapping.
MimConvert()	SrcImageId, DestImageId, ConversionType	Perform a color conversion.
MimConvolve()	SrcImageBufId, DestImageBufId, KernelBufId	Perform a general convolution operation.
MimCountDifference()	Src1ImageBufId, Src2ImageBufId, ImResultId	Count the number of pixels that differ in each image.
MimDeinterlace()*	ContextId, SrcImageArrayPtr, DstImageArrayPtr, SrcImageCount, DstImageCount, ControlFlag	Produce a sequence of deinterlaced images from a sequence of images acquired from an interlaced camera.
MimDilate()	SrcImageBufId, DestImageBufId, NbIteration, ProcMode	Perform a binary or grayscale dilation-type morphological operation.
MimDistance()	ScrImageBufId, DestImageBufId, DistanceTransform	Perform a distance transformation.
MimEdgeDetect()	SrcImageBufId, DestIntensityImageBufId, DestAngleImageBufId, KernelId, ControlFlag, Threshold	Perform a specific edge detection operation and produce a gradient intensity and/or gradient angle image.
MimErode()	SrcImageBufId, DestImageBufId, NbIteration, ProcMode	Perform an erosion-type morphological operation.
MimFindExtreme()	SrcImageBufId, ExtremeImResultId, ExtremeType	Find an image buffer's extremes (minimum and/or maximum pixel values)
MimFlip()	ScrImageId, DestImageId, Operation, OpFlag	Perform a horizontal or vertical image-flipping rotation.
MimFree()	ImResultId	Free an image processing result buffer.
MimGetResult()	ImResultId, ResultType, UserArrayPtr	Get values from an image processing result buffer.
MimGetResult1d()	ImResultId, OffEntry, NbEntries, ResultType, UserArrayPtr	Get values from a 1D region of an image processing result buffer.
MimHistogram()	SrcImageBufId, HistImResultId	Generate the intensity histogram of an image buffer.

* Available as of Processing Pack 1.

Image processing module (continued)

Commands	Command parameters	Description
MimHistogramEqualize()	SrcImageBufId, DestImageBufId, Method, Alpha, Min, Max	Perform a histogram equalization of an image.
MimInquire()	BufId, InquireType, UserVarPtr	Inquire about an image processing result buffer parameter setting.
MimLabel()	SrcImageBufId, DestImageBufId, ProcMode	Label objects in an image buffer.
MimLocateEvent()	SrcImageBufId, EventImResultId, Condition, CondLow, CondHigh	Find pixel coordinates or values that satisfies a specified condition.
MimLutMap()	SrcImageBufId, DestImageBufId, LutBufId	Perform a point-to-point LUT mapping operation.
MimMorphic()	SrcImageBufId, DestImageBufId, StructElemBufId, Operation, NBIteration, ProcMode	Perform a morphological transformation using a user-defined kernel.
MimOpen()	SrcImageBufId, DestImageBufId, NbIteration, ProcMode	Perform a binary or grayscale opening-type morphological operation.
MimPolarTransform()	SrcImageBufId, DestImageBufId, CenterPosX, CenterPosY, StartRadius, EndRadius, StartAngle, EndAngle, OperationMode, InterpolationMode, DestSizeXPtr, DestSizeYPtr	Perform a polar-to-rectangular or rectangular-to-polar transforms.
MimProject()	SrcImageBufId, ProjImResultId, ProjAngle	Project a 2D image into 1D.
MimRank()	SrcImageBufId, DestImageBufId, StructElemBufId, Rank, ProcMode	Perform a rank filter on the pixels in an image.
MimResize()	SrcImageBufId, DestImageBufId, ScaleFactorX, ScaleFactorY, InterpolationMode	Resize an image.
MimRotate()	SrcImageBufId, DestImageBufId, Angle, SrcCenX, SrcCenY, DstCenX, DstCenY, InterpolationMode	Rotate an image.
MimShift()	SrcImageBufId, DestImageBufId, BitsToShift	Perform a point-to-point bit shift.
MimStat()	SrcImageId, StatResultId, StatType, Condition, CondLow, CondHigh,	Calculate a variety of statistics on the source image.
MimThick()	SrcImageBufId, DestImageBufId NbIteration, ProcMode	Perform a binary or grayscale thickening operation on an image.
MimThin()	SrcImageBufId, DestImageBufId NbIteration, ProcMode	Perform a binary or grayscale thinning operation on an image.
MimTransform()	SrcImageRBufId, SrcImageIBufId, DestImageRBufId, DestImageIBufId TransformType, ControlFlag	Perform a Fast Fourier transform (FFT) or a Discrete Cosine transform (DCT).
MimTranslate()	SrcImageBufId, DestImageBufId, XDisplacement, YDisplacement, InterpolationMode	Translate an image in X and/or Y displacement.
MimWarp()	SrcImageId, DestImageId, WarpParam1Id, WarpParam2Id, OperationMode, InterpolationType	Perform a warping.
MimWatershed	SrcImageId, MarkerImageId, DestImageId, MinimumVariation, ControlFlag	Perform a watershed transformation.
MimZoneOfInfluence()	SrcImageBufId, DestImageBufId, OperationFlag	Perform a zone of influence detection.

Measurement module

Used to locate and measure edges or stripes within an image. Also used to take measurements between points, edges, or stripes. This module includes functions to save or restore markers (i.e., points, edges, or stripes).

Commands	Command parameters	Description
MmeasAllocContext()	SystemId, ControlFlag, ContextIdPtr	Allocate a measurement context.
MmeasAllocMarker()	SystemId, MarkerType, ControlFlag, MarkerIdPtr	Allocate a measurement marker.
MmeasAllocResult()	SystemId, ResultType, MeasResultIdPtr	Allocate a measurement result buffer.
MmeasCalculate()	ContextId, Marker1Id, Marker2Id, MeasResultId, MeasurementList	Calculate measurements between two markers.
MmeasControl()	ContextId, ControlType, Value	Control a measurement parameter setting.
MmeasDraw()	GraphContId, MarkerOrResultId, DestImageId, Operation, Index, ControlFlag	Draw features of specific markers or result occurrences in an image buffer.
MmeasFindMarker()	ContextId, ImageBufId, MarkerId, MeasurementList	Find a marker in an image and take the specified measurements.
MmeasFree()	MeasId	Free a measurement context, marker, or result buffer.
MmeasGetResult()	MarkerOrMeasResultId, ResultType, FirstResultArrayPtr, SecondResultArrayPtr	Get the results of measurements taken.
MmeasGetResultSingle()	MarkerOrMeasResultId, ResultType, FirstResultArrayPtr, SecondResultArrayPtr, ResultIndex	Get a single result from a multiple marker or its result buffer.
MmeasInquire()	MeasId, InquireType, FirstValuePtr, SecondValuePtr	Inquire about a measurement context, marker, or result buffer.
MmeasRestoreMarker()	FileName, SystemId, ControlFlag, MarkerIdPtr	Restore a marker from disk.
MmeasSaveMarker()	FileName, MarkerId, ControlFlag	Save a marker to disk.
MmeasSetMarker()	MarkerId, CharacteristicToSet, FirstValue, SecondValue	Set a marker characteristic parameter.

OCR module

Template-based character recognition module. This module includes control of character font definition, as well as font archiving and retrieving.

Commands	Command parameters	Description
MocrAllocFont()	SystemId, FontType, CharNumber, CharBoxSizeX, CharBoxSizeY, CharOffsetX, CharOffsetY, CharSizeX CharSizeY, CharThickness, StringLength, InitFlag, FontIdPtr	Allocate an OCR font buffer.
MocrAllocResult()	SystemId, InitFlag, OcrResultIdPtr	Allocate an OCR result buffer.
MocrCalibrateFont()	ImageBufId, FontId, String, TargetCharSizeXMin, TargetCharSizeXMax, TargetCharSizeXStep, TargetCharSizeYMin, TargetCharSizeYMax, TargetCharSizeYStep, Operation	Calibrate font character size to match a sample image.
MocrControl()	FontId, ControlToSet, Value	Control an OCR parameter setting.
MocrCopyFont()	ImageBufId, FontId, Operation, CharListString	Copy a font character to or from an image buffer.
MocrFree()	FontIdOrResultId	Free an OCR font or result buffer.
MocrGetResult()	OcrResultId, ResultToGet, ResultPtr	Read results from an OCR result buffer.
MocrHookFunction()	FontId, HookType, HookHandlerPtr, UserDataPtr	Hook a function to an event.
MocrImportFont()	FileName, FileFormat, Operation, CharListString, FontId	Import font data from file on disk.
MocrInquire()	FontId, InquireItem,UserVarPtr	Inquire about font character information.
MocrModifyFont()	FontId, Operation, ControlValue	Invert or resize a font to match the target image characters.
MocrPreprocess()	FontId, ControlFlag	Preprocess an OCR font context.
MocrReadString()	ImageBufId, FontId, OcrResultId	Read an unknown string from an image.
MocrRestoreFont()	FileName, Operation, SystemId, FontIdPtr	Restore a font from disk.
MocrSaveFont()	FileName, Operation, FontId	Save an existing font to disk.
MocrSetConstraint()	FontId, CharPos, CharPosType, CharValidString	Set character position constraints.
MocrVerifyString()	ImageBufId, FontId, String, OcrResultId	Verify a known string in an image.

Pattern matching module

Used to locate patterns in an image using normalized grayscale correlation (NGC). This module includes functions to define a pattern, control search strategy, and save and restore a pattern.

Commands	Command parameters	Description
MpatAllocAutoModel()	SystemId, SrcImageBufId, SizeX, SizeY, PosUncertaintyX, PostUncertaintyY, ModelType, Mode, ModelIdPtr	Automatically allocate unique pattern matching models of the specified type, from a source image.
MpatAllocModel()	SystemId, SrcImageBufId, OffX, OffY, SizeX, SizeY, ModelType, ModelIdPtr	Allocate a pattern matching model from a source image.
MpatAllocResult()	SystemId, NbEntries, PatResultIdPtr	Allocate a pattern matching result buffer.
MpatAllocRotatedModel()	SystemId, SrcModelId, Angle, InterpolationMode, ModelType, NewModelIdPtr	Rotate a pattern matching model.
MpatCopy()	ModelId, DestImageBufId, CopyMode	Copy a pattern matching model to an image buffer.
MpatDraw()	GraphContId, ModelOrResultId, DestImageId, Operation, Index, ControlFlag	Draw features of a specific model or result occurrences in an image buffer.
MpatFindModel()	ImageBufId, ModelId, PatResultId	Find a pattern matching model in the target image buffer.
MpatFindMultipleModel()	ImageBufId, ModelIdLst, PatResultIdst, NumModels, ExpFlag	Find multiple pattern matching models in the target image buffer.
MpatFree()	PatId	Free a pattern matching model or a result buffer.
MpatGetNumber()	PatResultId, CountPtr	Get the number of model occurrences in the target image.
MpatGetResult()	PatResultId, ResultType, UserArrayPtr	Get the pattern matching result values.
MpatInquire()	PatId, ParamToInquire, UserVarPtr	Inquire about the pattern matching model or the result buffer parameter setting.
MpatPreprocModel()	TypicalImageBufId, ModelId, Mode	Preprocess a pattern matching model.
MpatRead()	SystemId, FileHandle, ModelIdPtr	Read a pattern matching model from an open file.
MpatRestore()	SystemId, FileName, ModelIdPtr	Restore a pattern matching model from disk.
MpatSave()	FileName, ModelId	Save a pattern matching model to disk.
MpatSetAcceptance()	ModelId, AcceptanceThreshold	Set the acceptance level of a model.
MpatSetAccuracy()	ModelId, Accuracy	Set the positional accuracy of a model.
MpatSetAngle()	ModelId, ControlType, ControlValue	Set the angular search parameters of a model.
MpatSetCenter()	ModelId, OffX, OffY	Set the reference position of a model.
MpatSetCertainty()	ModelId, CertaintyThreshold	Set the certainty level of a model.
MpatSetDontCare()	ModelId, ImageBufId, OffX, OffY, Value	Set the “don’t care” pixels in a model.
MpatSetNumber()	ModelId, NbOccurences	Set the expected number of occurrences of a model.
MpatSetPosition()	ModelId, OffX, OffY, SizeX, SizeY	Set the search region of a model.
MpatSetSearchParameter()	PatId, Parameter, Value	Set the internal search parameters of a model.
MpatSetSpeed()	ModelId, SpeedFactor	Set search speed of a model.
MpatWrite()	FileHandle, ModelId	Write a pattern matching model to an open file.

String Reader module

Feature-based character recognition module. This module supports multiple user-defined grammar rules and multi-font definition in a single context.

Commands	Command parameters	Description
MstrAlloc()	SystemId, ContextType, ControlFlag, ObjectIdPtr	Allocates a String Reader context.
MstrAllocResult()	SystemId, ControlFlag, ObjectIdPtr	Allocate a String Reader result buffer.
MstrControl()	ContextOrResultID, Index, ControlType ControlValue	Control a String Reader context, a specific string model, a specific font, or a String Reader result buffer.
MstrDraw()	GraphContId, ContextOrResultID, DestImageId, Operation, Index, CharList ControlFlag	Draw specific features of the String Reader context or String Reader results.
MstrEditFont()	ContextId, FontIndex, Operation, OperationMode, Param1, Param2, Param3	Edit a specified font.
MstrFree()	ObjectId	Free a String Reader context or a String Reader result buffer.
MstrGetResult()	ResultId, Index, ResultType, ResultArrayPtr	Get the specified type of result(s) from a String Reader result buffer.
MstrInquire()	ContextOrResultId, Index, InquireType UserVarPtr,	Inquire information about a specified String Reader context, string model, font, or result buffer.
MstrPreprocess()	ContextId, ControlFlag	Preprocess a String Reader context.
MstrRead()	ContextId, TargetImageId, ResultId	Read strings from a target image.
MstrRestore()	Filename, SystemId, ControlFlag, ContextIdPtr	Restore a String Reader context from disk.
MstrSave()	FileName, ContextId, ControlFlag	Save a String Reader context to a file.
MstrSetConstraint()	ContextId, StringIndex, CharPos ConstraintType, CharList	Set character constraints.
MstrStream()	MemPtrOrFileName, SystemId, Operation, StreamType, Version ControlFlag, ObjectIdPtr, SizeByteVarPtr	Load, restore, or save a String Reader context from/to a file or a memory.

Thread module

Used for the allocation of MIL thread contexts and synchronization events. This module allows control over the created MIL thread contexts and events, inquire about various settings, and synchronize execution of multiple threads.

Commands	Command parameters	Description
MthrAlloc()	SystemId, ObjectType, ControlFlag ThreadFctPtr, UserPtr, ThreadOrEventId	Allocate a MIL thread context or event.
MthrControl()	ThreadOrEventId, ControlType, ControlValue	Control MIL thread context or MIL event settings.
MthrFree()	ThreadOrEventId	Free a MIL thread context or event.
MthrInquire()	ThreadOrEventId, InquireType, InquireValue	Inquire about a MIL thread context or event setting.
MthrWait()	ThreadOrEventId, WaitOption, State	Perform a wait operation on a MIL thread or event.

Programming Examples

Blob analysis (MIL example)

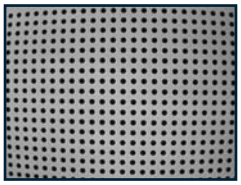
This program counts the number of objects in an image and locates the center of gravity of each objects.

“bolts.mim”

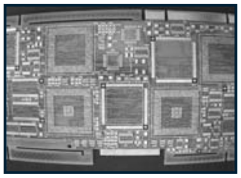
```
 #include <stdio.h>
  #include <mil.h>

 /* Maximum number of blobs and minimum area of blobs. */
 #define MAX_BLOBS                              100L
 #define MIN_BLOB_AREA                          50L
 #define IMAGE_FILE                             "bolts.mim"
 #define THRESHOLD_VALUE                        24L
 void main(void)
  {
 MIL_ID   MilApplication,                       /* Application identifier. */
          MilSystem,/* System identifier. */
          MilImage,                             /* Image buffer identifier. */
          BlobResult,                           /* Blob result buffer identifier. */
          FeatureList;                          /* Feature list identifier. */
 long     TotalBlobs,                           /* Total number of blobs. */
          CogX[MAX_BLOBS],                      /* X coordinate of center of gravity. */
          CogY[MAX_BLOBS]                       /* Y coordinate of center of gravity. */
          n;                                    /* Counter. */
 /* Allocate defaults. */
 MappAllocDefault(M_SETUP, &MilApplication, &MilSystem, M_NULL, M_NULL, &MilImage);

 /* Load source image into image buffer. */
 MbufLoad(IMAGE_FILE, MilImage);

 /* Binarize image. */
  MimBinarize(MilImage, MilImage, M_GREATER_OR_EQUAL, THRESHOLD_VALUE, M_NULL);

 /* Allocate a blob feature list and a result buffer. */
  MblobAllocFeatureList(&FeatureList);
  MblobAllocResult(&BlobResult);

 /* Enable features to be calculated. */
  MblobSelectFeature(FeatureList, M_AREA);
  MblobSelectFeature(FeatureList, M_CENTER_OF_GRAVITY_X);
  MblobSelectFeature(FeatureList, M_CENTER_OF_GRAVITY_Y);

 /* Calculate selected features for each blob. */
  MblobCalculate(MilImage, M_NULL, FeatureList, BlobResult);

 /* Exclude blobs whose area is too small. */
  MblobSelect(BlobResult, M_EXCLUDE, M_AREA, M_LESS_OR_EQUAL, MIN_BLOB_AREA, M_NULL);

 /* Get the total number of blobs and their center of gravity. */
  MblobGetNumber(BlobResult, &TotalBlobs);
  MblobGetResult(BlobResult, M_CENTER_OF_GRAVITY_X+M_TYPE_LONG, CogX);
  MblobGetResult(BlobResult, M_CENTER_OF_GRAVITY_Y+M_TYPE_LONG, CogY);

 /* Print the number of blobs and their center of gravity. */
  printf(“\nThere are %ld objects in the image,\n”, TotalBlobs);
  for(n=0; n< TotalBlobs; n++)
     printf(“Center of gravity: X=%ld, Y=%ld.\n”, CogX[n], CogY[n]);

/* Wait for a key press. */
printf("Press <Enter>.\n");
getchar();

 /* Free all allocations. */
 MblobFree(FeatureList);
 MblobFree(BlobResult);
 MappFreeDefault(MilApplication, MilSystem, M_NULL, M_NULL, MilImage);
 }
```

Calibration (MIL example)

This program illustrates camera calibration for a severe lens aberration using a calibration grid. An image acquired from the same camera is then compensated using the calibration results determined from the grid. Measurements are then performed on the calibrated image to find the width of the board in "real world" units (i.e., centimeters).



"CalGrid.mim"



"GenBoard.mim"

```
/* Regular includes. */
#include <mil.h>
#include <stdio.h>

/* Source image files specification. */
#define GRID_IMAGE_FILE                 "CalGrid.mim"
#define BOARD_IMAGE_FILE                "GenBoard.mim"

/* World description of the calibration grid. */
#define GRID_OFFSET_X                   0
#define GRID_OFFSET_Y                   0
#define GRID_OFFSET_Z                   0
#define GRID_ROW_SPACING                1
#define GRID_COLUMN_SPACING             1
#define GRID_ROW_NUMBER                 18
#define GRID_COLUMN_NUMBER              25

/* Measurement box specification */
#define MEAS_BOX_POS_X                  55
#define MEAS_BOX_POS_Y                  24
#define MEAS_BOX_WIDTH                  7
#define MEAS_BOX_HEIGHT                 425

/* Specification of the stripe constraints. */
#define APPROXIMATE_STRIPE_WIDTH        425
#define M_WIDTH_WEIGHT_FACTOR           98

/* Main application function */
void main()
{
   MIL_ID         MilApplication,            /* Application identifier. */
                    MilSystem,                 /* System identifier. */
                    MilImage,                  /* Image buffer identifier. */
                    MilCalibration,            /* Calibration identifier. */
                    MeasMarker,                /* Measurement marker identifier. */
   double WorldWidth;

  /* Allocate defaults */
  MappAllocDefault(M_SETUP, &MilApplication, &MilSystem, M_NULL, M_NULL, M_NULL);

  /* Restore an image of a grid grabbed with a camera with severe lens aberration into an automatically allocated image buffer.
*/
  MbufRestore(GRID_IMAGE_FILE, MilSystem, &MilImage);

  /* Allocate a camera calibration object. */
  McalAlloc(M_DEFAULT, M_DEFAULT, &MilCalibration);

  /* Calibrate the camera with the image of the grid
   * and its world description.
   */
  McalGrid(MilCalibration, MilImage,
      GRID_OFFSET_X, GRID_OFFSET_Y, GRID_OFFSET_Z, GRID_ROW_NUMBER, GRID_COLUMN_NUMBER,
      GRID_ROW_SPACING, GRID_COLUMN_SPACING, M_DEFAULT, M_DEFAULT);
```

Calibration (continued)

```
  /* Load an image of a board grabbed with a camera with severe lens aberration and associate the calibration to the image. */
  MbufLoad(BOARD_IMAGE_FILE, MilImage);
  McalAssociate(MilCalibration, MilImage, M_DEFAULT);

  /* Allocate a measurement marker to perform measurement on the calibrated image in "real world" unit. */
  MmeasAllocMarker(MilSystem, M_STRIPE, M_DEFAULT, &MeasMarker);

  /* Set the marker measurement box. */
  MmeasSetMarker(MeasMarker, M_BOX_ORIGIN, MEAS_BOX_POS_X, MEAS_BOX_POS_Y);
  MmeasSetMarker(MeasMarker, M_BOX_SIZE, MEAS_BOX_WIDTH, MEAS_BOX_HEIGHT);

  /* Set marker orientation. */
  MmeasSetMarker(MeasMarker, M_ORIENTATION, M_HORIZONTAL, M_NULL);

  /* Set marker approximative width and the associated weight factor. */
  MmeasSetMarker(MeasMarker, M_WIDTH, APPROXIMATE_STRIPE_WIDTH, M_NULL);
  MmeasSetMarker(MeasMarker, M_WEIGHT_FACTOR+M_WIDTH, M_WIDTH_WEIGHT_FACTOR, M_NULL);

  /* Find the stripe (2 board edges) in the calibrated image and measure its width in "real world" unit.
   */
  MmeasFindMarker(M_DEFAULT, MilImage, MeasMarker, M_WIDTH);

  /* Get the world width of the marker. */
  MmeasGetResult(MeasMarker, M_WIDTH, &WorldWidth, M_NULL);

  /* Pause to show the measurement result. */
  printf("The board width is %8.4lf centimeters.\n", WorldWidth);
  printf("Press <Enter> to end.\n\n");
  getchar();

  /* Free all allocations */
  MmeasFree(MeasMarker);
  McalFree(MilCalibration);
  MbufFree(MilImage);
  MappFreeDefault(MilApplication, MilSystem, M_NULL, M_NULL, M_NULL);
}
```

Camera auto-focus (MIL example)

This program illustrates the use of the auto-focus tool to adjust camera focus by way of a motorized lens.

```
/* Regular includes. */
#include <stdio.h>
#include <mil.h>
#include <stdlib.h>

/* Lens characteristics */
#define FOCUS_MAX_NB_POSITIONS          256
#define FOCUS_MIN_POSITION              0
#define FOCUS_MAX_POSITION              255
#define FOCUS_START_POSITION            0

/* Autofocus search properties */
#define FOCUS_MAX_POSITION_VARIATION  32
#define FOCUS_MODE                      M_SMART_SCAN
#define FOCUS_SENSITIVITY               1

/* Autofocus hook function that is responsible to move the lens */
long MFTYPE MoveLensFunction(long HookType, long position, void MPTYPE *UserDataPtr);

/* Main application function                          */
/***************************************************************/
void main(void)
{
 MIL_ID MilApplication,                    /* Application identifier. */
        MilSystem,                         /* System identifier. */
        MilDisplay,                        /* Display identifier. */
        MilDigitizer,                      /* Digitizer identifier. */
        MilImage;                          /* Image buffer identifier. */
  long    FocusPos;                        /* Best focus position */

  /* Allocate defaults */
  MappAllocDefault(M_SETUP, &MilApplication, &MilSystem, &MilDisplay, &MilDigitizer, &MilImage);

  /* Grab continuously. */
  MdigGrabContinuous(MilDigitizer, MilImage);

  /* Pause to show the original image. */
  printf("An autofocus operation will be performed.\n");
  printf("Press <Enter> to continue.\n");
  getchar();
  printf("Autofocusing...\n");

  /* Perform autofocus by calling the MoveLensFunction iteratively */
  MdigFocus (MilDigitizer,
           MilImage,
           M_DEFAULT,
           MoveLensFunction,
           M_NULL,
           FOCUS_MIN_POSITION,
           FOCUS_START_POSITION,
           FOCUS_MAX_POSITION,
           FOCUS_MAX_POSITION_VARIATION,
           FOCUS_MODE + FOCUS_SENSITIVITY, &FocusPos);

  /* Print the best focus position and number of iterations*/
  printf("The best focus position is %d.\n", FocusPos);
```

Camera auto-focus (continued)

```
   printf("Press <Enter> to end.\n");
   getchar();

   /* Free all allocations */
   MappFreeDefault(MilApplication, MilSystem, MilDisplay, MilDigitizer, MilImage);
}

/* Autofocus hook function that is responsible to move the lens */
/***************************************************************/
long MFTYPE MoveLensFunction(long HookType, long position, void MPTYPE *UserDataPtr)
{
   /* If focus position must be changed */
   if(HookType == M_CHANGE)
     {
     /* Move the camera lens to the specified
         position using the appropriate interface (e.g. serial port).
     */
     MoveLens(position);
     }

   return 0;
}
```

Capture and display a video sequence (MIL/MIL-Lite example)

This program grabs a sequence of images and plays it back continuously.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <mil.h>

#define IMAGE_WIDTH             640
#define IMAGE_HEIGHT            480
#define NBGRAB  8               /* Number of image buffers in the sequence. */

void main(void)
{
  MIL_ID MilApplication,        /* Application identifier. */
         MilSystem,             /* System identifier. */
         MilDigitizer,          /* Digitizer identifier. */
         MilDisplay,            /* Display identifier. */
         MilImage[NBGRAB],      /* Number of image buffers in the sequence. */
         MilImageDisp;          /* Display image buffer identifier. */

  long n;

  /* Allocate defaults. */
  MappAllocDefault(M_SETUP, &MilApplication, &MilSystem, &MilDisplay, &MilDigitizer, M_NULL);

  /* Allocate sequence storage image buffers. */
  for (n=0; n<NBGRAB; n++)
  {
  MbufAlloc2d(MilSystem, IMAGE_WIDTH, IMAGE_HEIGHT, 8L+M_UNSIGNED, M_IMAGE+M_GRAB, &MilImage[n]);
  }

  /* Allocate a display image buffer. */
  MbufAlloc2d(MilSystem, IMAGE_WIDTH, IMAGE_HEIGHT, 8L+M_UNSIGNED,
            M_IMAGE+M_GRAB+M_PROC+M_DISP, &MilImageDisp);

  /* Display activation. */
  MbufClear(MilImageDisp, 0x0);
  MdispSelect(MilDisplay, MilImageDisp);

  /* Print a message. */
  printf(“Press enter to record the sequence.\n”);
  getchar();

  /* Grab the sequence. */
  for (n=0; n<NBGRAB; n++)
    {
    MdigGrab(MilDigitizer, MilImage[n]);
    }

 /* Play the sequence until a key is pressed. */
 while( !kbhit() )
   {
    /* Play the sequence once. */
    for (n=0; n<NBGRAB; n++)
      {
       /* Copy one image to the screen. */
       MbufCopy(MilImage[n],MilImageDisp);
       }
   }

 /* Free image buffers. */
 MbufFree(MilImageDisp);
 for (n=0; n<NBGRAB; n++)
   {
   MbufFree(MilImage[n]);
   }

/* Free defaults. */
MappFreeDefault(MilApplication, MilSystem, MilDisplay, MilDigitizer, M_NULL);
}
```

Note: Playback can be synchronized with the display.

Code Reader (MIL example)

This program decodes a DataMatrix code. The string is read and then printed to the screen.

"excode.mim"

```
#include <stdio.h>
#include <string.h>
#include <mil.h>

/* Target image character specifications. */
#define CHAR_IMAGE_FILE          "excode.mim"

/* Maximum length of the string to read or draw (null terminated) */
#define STRING_LENGTH                           64L

/* Threshold value */
#define THRESHOLD_VALUE                         128L

void main(void)
{
  MIL_ID MilApplication,                        /* Application identifier. */
         MilSystem,                             /* System identifier. */
         MilDisplay,                            /* Display identifier. */
         MilImage,                              /* Image buffer identifier. */
         MatrixCode;                            /* DataMatrix code identifier */

  char   ResultString[STRING_LENGTH];           /* Array of characters read. */

  /* Allocate defaults */
  MappAllocDefault(M_SETUP, &MilApplication, &MilSystem, &MilDisplay, M_NULL, M_NULL);

  /* Restore source image into an automatically allocated image buffer. */
  MbufRestore(CHAR_IMAGE_FILE, MilSystem, &MilImage);

  /* Display the image buffer. */
  MdispSelect(MilDisplay, MilImage);

  /* Allocate CODE object */
  McodeAlloc(MilSystem, M_DATAMATRIX, M_DEFAULT, &MatrixCode);

  /* Set threshold value */
  McodeControl(MatrixCode, M_THRESHOLD, THRESHOLD_VALUE);

  /* Pause to show the original image. */
  printf("This program will decode a DataMatrix code.\n");
  printf("Press <Enter> to continue.\n");
  getchar();

  /* Read code from image */
  McodeRead(MatrixCode, MilImage, M_DEFAULT);

  /* Get decoded string */
  McodeGetResult(MatrixCode, M_STRING, ResultString);

  printf("The decoded string is : %s\n", ResultString);
  printf("Press <Enter> to end.\n");
  getchar();

  /* Free all allocations. */
  McodeFree(MatrixCode);
  MbufFree(MilImage);
  MappFreeDefault(MilApplication, MilSystem, MilDisplay, M_NULL, M_NULL);
}
```

Digitizer allocation and control (MIL/MIL-Lite example)

This program illustrates continuous image capture to display.

```
#include <stdio.h>
#include <mil.h>

void main(void)
{
  MIL_ID MilApplication,              /* Application identifier. */
              MilSystem,               /* System identifier. */
              MilDisplay,              /* Display identifier. */
              MilCamera,               /* Camera identifier. */
              MilImage;                /* Image buffer identifier. */

  /* Allocate defaults. */
  MappAllocDefault(M_SETUP, &MilApplication, &MilSystem, &MilDisplay, &MilCamera, &MilImage);

  /* Grab continuously. */
  MdigGrabContinuous(MilCamera, MilImage);

  /* When a key is pressed, halt. */
  printf(“Continuous grab in progress. Adjust your camera and\n”);
  printf(“press <Enter> to stop grabbing.”);
  getchar();

  /* Stop continuous grab. */
  MdigHalt(MilCamera);
  printf(“\nDisplaying the last grabbed image.\n”);
  printf(“Press <Enter> to end.”);
  getchar(

  /* Release defaults. */
  MappFreeDefault(MilApplication, MilSystem, MilDisplay, MilCamera, MilImage);
}
```

Displaying a MIL buffer under Windows (MIL/MIL-Lite example)

This program could form the core of a window-based MIL application. It displays a MIL image buffer in a user specified window. The MIL buffer is initialized with text drawn using MIL graphic functions.

Note: For simplicity, the program entry point is not included.

```
void MilWindowsApplication(HWND UserWindowHandle)
{
  MIL_ID  MilApplication,             /* MIL Application identifier. */
          MilSystem,                  /* MIL System identifier. */
          MilDisplay,                 /* MIL Display identifier. */
          MilImage;                   /* MIL Image buffer identifier. */

  /* Allocate defaults. */
  MappAllocDefault(M_SETUP, &MilApplication, &MilSystem, &MilDisplay, M_NULL, &MilImage);

  /* Select the image buffer to be displayed into the specified user window. */
  MdispSelectWindow(MilDisplay, MilImage, UserWindowHandle);
```

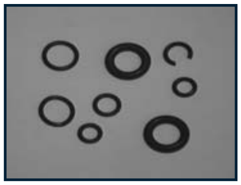
Digitizer allocation and control (MIL/MIL-Lite example)

This program illustrates continuous image capture to display.

```
#include <stdio.h>
#include <mil.h>

void main(void)
{
  MIL_ID MilApplication,              /* Application identifier. */
          MilSystem,                  /* System identifier. */
          MilDisplay,                 /* Display identifier. */
          MilCamera,                  /* Camera identifier. */
          MilImage;                   /* Image buffer identifier. */

  /* Allocate defaults. */
  MappAllocDefault(M_SETUP, &MilApplication, &MilSystem, &MilDisplay, &MilCamera, &MilImage);

  /* Grab continuously. */
  MdigGrabContinuous(MilCamera, MilImage);

  /* When a key is pressed, halt. */
  printf(“Continuous grab in progress. Adjust your camera and\n”);
  printf(“press <Enter> to stop grabbing.”);
  getchar();

  /* Stop continuous grab. */
  MdigHalt(MilCamera);
  printf(“\nDisplaying the last grabbed image.\n”);
  printf(“Press <Enter> to end.”);
  getchar(

  /* Release defaults. */
  MappFreeDefault(MilApplication, MilSystem, MilDisplay, MilCamera, MilImage);
}


  /* Print a string in the image buffer using MIL. */
  MgraText(M_DEFAULT, MilImage, 176L, 210L, “ —————————————- “);
  MgraText(M_DEFAULT, MilImage, 176L, 235L, “ Welcome to MIL !!! “);
  MgraText(M_DEFAULT, MilImage, 176L, 260L, “ —————————————- “);
```

Edge Finder (MIL example)

This program illustrates edge extraction with exclusion. Edges are located in the target image, some results are excluded based predefined criteria (edge features), final results (drawn edges and total count) are printed to screen.



"Seals.mim"

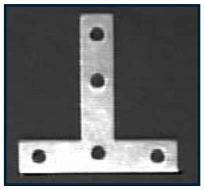
```
/* Regular includes. */
#include <mil.h>
#include <stdio.h>
#include <conio.h>

/* Source MIL image file specifications. */
#define CONTOUR_IMAGE              M_IMAGE_PATH MIL_TEXT("Seals.mim")
#define CONTOUR_MAX_RESULTS        100L
#define CONTOUR_MAXIMUM_ELONGATION  0.8

/*****************************************************************************
 Main.
 *****************************************************************************/
void main(void)
  {
  MIL_ID MilApplication,                    /* Application identifier.   */
         MilSystem,                         /* System Identifier.        */
         MilImage,                          /* Image buffer identifier.  */
         MilEdgeContext,                    /* Edge context              */
         MilResult;                         /* Result identifier.        */

  long   NumResults   = 0L,                 /* Number of results found.  */
         NumEdgeFound = 0L;                 /* Number of edges found.    */
  double MeanFeretDiameter[CONTOUR_MAX_RESULTS]; /* Edge mean Feret diameter. */
  int    i;                                 /* Loop variable             */

  /* Allocate defaults. */
  MappAllocDefault(M_SETUP, &MilApplication, &MilSystem, M_NULL, M_NULL, M_NULL);

  /* Load Restore the image and display it */
  MbufRestore(CONTOUR_IMAGE, MilSystem, &MilImage);

  /* Allocate a edge finder context. */
  MedgeAlloc(MilSystem, M_CONTOUR, M_DEFAULT, &MilEdgeContext);

  /* Allocate a result buffer. */
  MedgeAllocResult(MilSystem, M_DEFAULT, &MilResult);

  /* Enable features to compute */
  MedgeControl(MilEdgeContext, M_FERET_MEAN_DIAMETER+M_SORT1_DOWN, M_ENABLE);
  MedgeControl(MilEdgeContext, M_MOMENT_ELONGATION,                M_ENABLE);

  /* Calculate edges */
  MedgeCalculate(MilEdgeContext, MilImage, M_NULL, M_NULL, M_NULL, MilResult, M_DEFAULT);

  /* Get the number of found edges. */
  MedgeGetResult(MilResult, M_DEFAULT, M_NUMBER_OF_CHAINS+M_TYPE_LONG, &NumEdgeFound, M_NULL);

  /* Exclude elongated edges */
  MedgeSelect(MilResult, M_EXCLUDE, M_MOMENT_ELONGATION, M_LESS, CONTOUR_MAXIMUM_ELONGATION, M_NULL);

  /* Get the number of edges found. */
  MedgeGetResult(MilResult, M_DEFAULT, M_NUMBER_OF_CHAINS+M_TYPE_LONG, &NumResults, M_NULL);
```

Edge Finder (continued)

```
/* If edges have been found */
if ( (NumResults >= 1) && (NumResults <= CONTOUR_MAX_RESULTS) )
  {
  /* Exclude inner chains */
  MedgeSelect(MilResult, M_EXCLUDE, M_INCLUDED_EDGES, M_INSIDE_BOX, M_NULL, M_NULL);

  /* Get the number of edges found. */
  MedgeGetResult(MilResult, M_DEFAULT, M_NUMBER_OF_CHAINS+M_TYPE_LONG, &NumResults, M_NULL);

  /* Get the mean Feret diameters of the outer edges */
  MedgeGetResult(MilResult, M_DEFAULT, M_FERET_MEAN_DIAMETER, &MeanFeretDiameter, M_NULL);

  /* Now print the mean diameter of each outer edge. */
  printf("The mean diameter for each outer edge is:\n\n");
  printf("Index   Mean diameter \n");
  for (i=0; i<NumResults; i++)
    {
    printf("%-11d%-13.2f%\n", i, MeanFeretDiameter[i]);
    }
  }

/* Wait for a key press. */
printf("Press <Enter> to end.\n");
getch();

/* Free MIL objects. */
MbufFree(MilImage);
MedgeFree(MilEdgeContext);
MedgeFree(MilResult);

/* Free defaults. */
MappFreeDefault(MilApplication, MilSystem, M_NULL, M_NULL, M_NULL);
}
```

Geometric Model Finder (MIL example)

This program defines a sngle model and searches for the model in a target image based on geometric features.


"SimpleModel.mim"

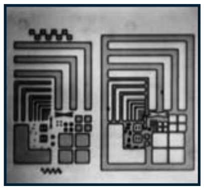
```
#include <stdio.h>
#include <mil.h>

#define MODEL_IMAGE                    "SingleModel.mim"
#define MODEL_TARGET_IMAGE             "SimpleTarget.mim"
#define MODEL_SEARCH_SPEED             M_VERY_HIGH
#define MODEL_OFFSETX                  176L
#define MODEL_OFFSETY                  136L
#define MODEL_SIZEX                    128L
#define MODEL_SIZEY                    128L
#define MODEL_MAX_OCCURRENCES          16L
#define MODEL_DRAW_COLOR               M_RGB888(255, 0, 0) /* Red */

void main(void)
  {
  MIL_ID MilApplication,                          /* Application identifier. */
         MilSystem,                               /* System Identifier. */
         MilDisplay,                              /* Display identifier. */
         MilImage,                                /* Image buffer identifier. */
         MilOverlayImage,                         /* Overlay image. */
         MilSearchContext,                        /* Search context */
         MilResult;                               /* Result identifier. */

  long    Model[MODEL_MAX_OCCURRENCES],           /* Model index. */
         ModelDrawColor = MODEL_DRAW_COLOR;      /* Model draw color */

  long    TransparentColor,                       /* Overlay clear color. */
         NumResults  = 0L;                        /* Number of results found. */

  double Score[MODEL_MAX_OCCURRENCES],            /* Model correlation score. */
         XPosition[MODEL_MAX_OCCURRENCES],        /* Model X position. */
         YPosition[MODEL_MAX_OCCURRENCES],        /* Model Y position. */
         Angle[MODEL_MAX_OCCURRENCES],            /* Model occurrence angle. */
         Scale[MODEL_MAX_OCCURRENCES],            /* Model occurrence scale. */
         Time = 0.0;                             /* Bench variable. */

  int     i;                                     /* Loop variable */

  /* Allocate defaults. */
  MappAllocDefault(M_SETUP, &MilApplication, &MilSystem, &MilDisplay, M_NULL, M_NULL);

  /* Load Restore the model image and display it */
  MbufRestore(MODEL_IMAGE, MilSystem, &MilImage);
  MdispSelect(MilDisplay, MilImage);

  /* Prepare for overlay annotations. */
  MdispControl(MilDisplay, M_WINDOW_OVR_WRITE, M_ENABLE);
  MdispInquire(MilDisplay, M_WINDOW_OVR_BUF_ID, &MilOverlayImage);
  MdispInquire(MilDisplay, M_KEY_COLOR, &TransparentColor);
  if (MbufInquire(MilOverlayImage, M_SIZE_BAND, M_NULL) == 1)
    ModelDrawColor = 0xFF;

  /* Allocate a geometric model finder. */
  MmodAlloc(MilSystem, M_GEOMETRIC, M_DEFAULT, &MilSearchContext);

  /* Allocate a result buffer. */
  MmodAllocResult(MilSystem, M_DEFAULT, &MilResult);

  /* Define the model */
  MmodDefine(MilSearchContext, M_IMAGE, MilImage,
         MODEL_OFFSETX, MODEL_OFFSETY, MODEL_SIZEX, MODEL_SIZEY);

  /* Set the search speed */
  MmodControl(MilSearchContext, M_CONTEXT, M_SPEED, MODEL_SEARCH_SPEED);
  /* Preprocess the search context. */
  MmodPreprocess(MilSearchContext, M_DEFAULT);
```

Geometric Model Finder (continued)

```
/* Draw box and position in the source image to show the model. */
MgraColor(M_DEFAULT, ModelDrawColor);
MmodDraw(M_DEFAULT, MilSearchContext, MilOverlayImage,
      M_DRAW_BOX+M_DRAW_POSITION, 0, M_ORIGINAL);

/* Pause to show the model. */
printf("A model finder context was defined with the model in the displayed image.\n");
printf("Press <Enter> to continue.\n");
getchar();

/* Clear the overlay image. */
MbufClear(MilOverlayImage, TransparentColor);

/* Load the single model target image. */
MbufLoad(MODEL_TARGET_IMAGE, MilImage);

/* Dummy first find for better function timing accuracy (model cache effet,...). */
MmodFind(MilSearchContext, MilImage, MilResult);

/* Search for the model. */
MappTimer(M_TIMER_RESET, M_NULL);
MmodFind(MilSearchContext, MilImage, MilResult);
MappTimer(M_TIMER_READ, &Time);

/* Get the number of models found. */
MmodGetResult(MilResult, M_DEFAULT, M_NUMBER+M_TYPE_LONG, &NumResults);

/* If a model was found above the acceptance threshold. */
if ( (NumResults >= 1) && (NumResults <= MODEL_MAX_OCCURRENCES) )
  {
  /* Get the results for each model. */
  MmodGetResult(MilResult, M_DEFAULT, M_INDEX+M_TYPE_LONG, Model);
  MmodGetResult(MilResult, M_DEFAULT, M_POSITION_X, XPosition);
  MmodGetResult(MilResult, M_DEFAULT, M_POSITION_Y, YPosition);
  MmodGetResult(MilResult, M_DEFAULT, M_ANGLE, Angle);
  MmodGetResult(MilResult, M_DEFAULT, M_SCALE, Scale);
  MmodGetResult(MilResult, M_DEFAULT, M_SCORE, Score);

  /* Print the results for each model found. */
  printf("The model finder context was used to find the model in the target image.\n\n");
  printf("Result   Model   X Position   Y Position   Angle   Scale   Score\n\n");
  for (i=0; i<NumResults; i++)
    {
    printf("%-9d%-8d%-13.2f%-13.2f%-8.2f%-8.2f%-5.2f%%\n",
        i, Model[i], XPosition[i], YPosition[i], Angle[i], Scale[i], Score[i]);
    }
  printf("\nThe search time is %.1f ms\n\n", Time*1000.0);

  /* Draw edges, position and box over the occurrences that were found. */
  for (i=0; i<NumResults; i++)
    {
    MgraColor(M_DEFAULT, ModelDrawColor);
    MmodDraw(M_DEFAULT,  MilResult, MilOverlayImage,
          M_DRAW_EDGES+M_DRAW_BOX+M_DRAW_POSITION, i, M_DEFAULT);
    }
  }
else
  {
  printf("The model was not found or the number of models found is greater than\n");
  printf("the specified maximum number of occurrence !\n\n");
  }

/* Wait for a key press. */
printf("Press <Enter>.\n");
getchar();

/* Free MIL objects. */
MbufFree(MilImage);
MmodFree(MilSearchContext);
MmodFree(MilResult);

/* Free defaults. */
MappFreeDefault(MilApplication, MilSystem, MilDisplay, M_NULL, M_NULL);
}
```

Image processing - convolution (MIL example)

This program loads an image and performs a 3x3 custom convolution operation (smoothing) on it.

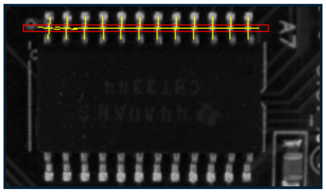


“wafer.mim”

```
#include <stdio.h>
#include <mil.h>

/* Target MIL image file specifications. */
#define IMAGE_FILE                           “wafer.mim”
#define IMAGE_WIDTH                          512L
#define IMAGE_HEIGHT                         480L

/* Kernel information. */
#define KERNEL_WIDTH                         3L
#define KERNEL_HEIGHT                        3L
#define KERNEL_DEPTH                         8L

/* Average kernel information data definition. */
unsigned char   KernelData[KERNEL_HEIGHT] [KERNEL_WIDTH] =
                { {1, 2, 1},
                {2, 4, 2},
                {1, 2, 1}
               };
void main(void)
{
  MIL_ID  MilApplication,                     /* Application identifier. */
          MilSystem,                          /* System identifier. */
          MilDisplay,                         /* Display identifier. */
          MilImage,                           /* Image buffer identifier. */
          MilSubImage,                        /* Sub-image buffer identifier. */
          MilKernel;                          /* Custom kernel identifier. */

  /* Allocate defaults. */
  MappAllocDefault(M_SETUP, &MilApplication, &MilSystem, &MilDisplay, M_NULL, &MilImage);

  /* Restrict the region to be processed to the image size. */
  MbufChild2d(MilImage, 0L, 0L, IMAGE_WIDTH, IMAGE_HEIGHT, &MilSubImage);

  /* Load source image into an image buffer. */
  MbufLoad(IMAGE_FILE, MilSubImage);

  /* Pause to show the original image. */
  printf(“This program does a convolution on the displayed image.\n”);
  printf(“It uses a custom smoothing kernel.\n”);
  printf(“Press <Enter> to continue.”);
  getchar();

  /* Allocate a MIL kernel. */
  MbufAlloc2d(M_DEFAULT, KERNEL_HEIGHT, KERNEL_WIDTH, KERNEL_DEPTH+M_UNSIGNED,  M_KERNEL, &MilKernel);

  /* Put the custom data in it. */
  MbufPut(MilKernel, KernelData);

  /* Set a normalization (divide) factor to have a kernel with a sum equal to one. */
  MbufControlNeighborhood(MilKernel, M_NORMALIZATION_FACTOR, 16L);

  /* Convolve the image using the kernel. */
  MimConvolve(MilSubImage, MilSubImage, MilKernel);

  /* Pause to show the result. */
  printf(“\n”);
  printf(“The original image was smoothed using a custom kernel.\n”);
  printf(“Press <Enter> to terminate.”);
  getchar();

  /* Free all allocations. */
  MbufFree(MilKernel);
  MbufFree(MilSubImage);
  MappFreeDefault(MilApplication, MilSystem, MilDisplay, M_NULL, MilImage);
}
```

Measurement (MIL example)

This program measures the position, width and angle of a stripe in an image, and marks its center and edges.



"chip.mim"

```
/* Regular includes. */
#include <stdio.h>
#include <mil.h>

/* Source MIL image file specification. */
#define IMAGE_FILE                          "chip.mim"

/* Processing region specification. */
#define MEAS_BOX_WIDTH                      128
#define MEAS_BOX_HEIGHT                     100
#define MEAS_BOX _POS_X                     166
#define MEAS_BOX _POS_Y                     130

/* Target stripe typical specifications. */
#define STRIPE_POLARITY_LEFT                M_POSITIVE
#define STRIPE_POLARITY_RIGHT               M_NEGATIVE
#define STRIPE_WIDTH                        45L
#define STRIPE_WIDTH_VARIATION              10L

/* Size and color of the cross to mark the positions. */
#define CROSS_SIZE                          10L
#define CROSS_COLOR                         240L

/* Utility functions prototypes */
void DrawCross(MIL_ID ImageId, double CenterX, double CenterY, long Color);

/* Main application function */
void main(void)
{
  MIL_ID MilApplication,                    /* Application identifier */
         MilSystem,                         /* System identifier. */
         MilDisplay,                        /* Display identifier. */
         MilImage,                          /* Image buffer identifier. */
         StripeMarker;                      /* Stripe marker identifier. */
  double  StripeCenterX                     /* Stripe X center position. */
         StripeCenterY                      /* Stripe Y center position. */
         StripeWidth,                       /* Stripe width. */
         StripeAngle                        /* Stripe angle. */
         StripeScore,                       /* Stripe Score. */
         StripeFirstEdgeX,                  /* Stripe left edge X position. */
         StripeFirstEdgeY,                  /* Stripe left edge Y position. */
         StripeSecondEdgeX,                 /* Stripe right edge X position. */
         StripeSecondEdgeY;                 /* Stripe right edge Y position. */

  /* Allocate defaults */
   MappAllocDefault(M_SETUP, &MilApplication, &MilSystem, &MilDisplay, M_NULL, &MilImage);

  /* Read the source image */
   MbufLoad(IMAGE_FILE, MilImage);

  /* Draw the contour of the measurement box */
   MgraRect(M_DEFAULT, MilImage, MEAS_BOX_POS_X-1, MEAS_BOX_POS_Y-1,
           MEAS_BOX_POS_X+MEAS_BOX_WIDTH+1, MEAS_BOX_POS_Y+MEAS_BOX_HEIGHT+1);

  /* Pause to show the original image. */
   printf("This program will determine the position, width and angle of the\n");
   printf("stripe in the highlighted box and mark its center and edges.\n");
   printf("Press <Enter> to continue.\n\n");
   getchar();
```

Measurement (continued)

```
/* Read the source image again to remove previously drawn rectangle */
   MbufLoad(IMAGE_FILE, MilImage);
/* Allocate a stripe marker */
 MmeasAllocMarker(M_DEFAULT, M_STRIPE, M_DEFAULT, &StripeMarker);

/* Specify the stripe approximative definition */
 MmeasSetMarker(StripeMarker, M_POLARITY, STRIPE_POLARITY_LEFT, STRIPE_POLARITY_RIGHT);
 MmeasSetMarker(StripeMarker, M_WIDTH, STRIPE_WIDTH, M_NULL);
 MmeasSetMarker(StripeMarker, M_WIDTH_VARIATION, STRIPE_WIDTH_VARIATION, M_NULL);
 MmeasSetMarker(StripeMarker, M_BOX_ANGLE_MODE, M_ENABLE, M_NULL);

/* Specify the search box size. */
 MmeasSetMarker(StripeMarker, M_BOX_ORIGIN, MEAS_BOX_POS_X, MEAS_BOX_POS_Y);
 MmeasSetMarker(StripeMarker, M_BOX_SIZE, MEAS_BOX_WIDTH, MEAS_BOX_HEIGHT);

/* Find the stripe and measure its width and angle. */
 MmeasFindMarker(M_DEFAULT, MilImage , StripeMarker, M_DEFAULT);

/* Get the stripe position, width and angle. */
 MmeasGetResult(StripeMarker, M_POSITION, &StripeCenterX, &StripeCenterY);
 MmeasGetResult(StripeMarker, M_POSITION+M_EDGE_FIRST, &StripeFirstEdgeX, &StripeFirstEdgeY);
 MmeasGetResult(StripeMarker, M_POSITION+M_EDGE_SECOND, &StripeSecondEdgeX, &StripeSecondEdgeY);
 MmeasGetResult(StripeMarker, M_WIDTH, &StripeWidth, M_NULL);
 MmeasGetResult(StripeMarker, M_ANGLE, &StripeAngle, M_NULL);
 MmeasGetResult(StripeMarker, M_SCORE, &StripeScore, M_NULL);

/* Draw a cross on the center, left edge and right edge of the found stripe. */
 DrawCross(MilImage, StripeCenterX, StripeCenterY, CROSS_COLOR);
 DrawCross(MilImage, StripeFirstEdgeX, StripeFirstEdgeY, CROSS_COLOR);
 DrawCross(MilImage, StripeSecondEdgeX, StripeSecondEdgeY, CROSS_COLOR);

/* Draw the contour of the measurement box */
 MgraRect(M_DEFAULT, MilImage, MEAS_BOX_POS_X-1, MEAS_BOX_POS_Y-1, MEAS_BOX_POS_X+MEAS_BOX_WIDTH+1,
             MEAS_BOX_POS_Y+MEAS_BOX_HEIGHT+1);

/* Print the result. */
 printf("The stripe in the box is at position %.2f,%.2f and\n", StripeCenterX, StripeCenterY);
 printf("is %.2f pixels wide with an angle of %.2f degrees.\n", StripeWidth, StripeAngle);
 printf("Its center and edges have been marked.\n");
 printf("Press <Enter> to continue.\n\n");
 getchar();

/* Free all allocations. */
 MmeasFree(StripeMarker);
 MappFreeDefault(MilApplication, MilSystem, MilDisplay, M_NULL, MilImage);
}

/* Draw a cross at the specified position. */
void DrawCross(MIL_ID ImageId, double CenterX, double CenterY, long Color)

{
   MgraColor(M_DEFAULT,Color);
   MgraLine(M_DEFAULT, ImageId,(long)(CenterX+.5)-(CROSS_SIZE/2), (long)(CenterY+.5),
             (long)(CenterX+.5)+(CROSS_SIZE/2), (long)(CenterY+.5));
   MgraLine(M_DEFAULT, ImageId, (long)(CenterX+.5), (long)(CenterY+.5)-(CROSS_SIZE/2), (long)(CenterX+.5),
             (long)(CenterY+.5)+(CROSS_SIZE/2));
}
```

Multi-buffered image capture and processing (MIL example)

This program shows the use of the MdigProcess() function to perform real-time image capture and processing.

```
#include <mil.h>
#include <conio.h>
#include <stdlib.h>

 /* Number of images in the buffering grab queue.    Generally, increasing this number gives better real-time grab.  */
#define BUFFERING_SIZE_MAX 20

/* User's processing function prototype. */
long MFTYPE ProcessingFunction(long HookType, MIL_ID HookId, void MPTYPE *HookDataPtr);

/* User's processing function hook data structure. */
typedef struct
  {
  MIL_ID  MilImageDisp;
  long    ProcessedImageCount;
  } HookDataStruct;

/* Main function. */
/* ---------------*/

void main(void)
{
  MIL_ID MilApplication;
  MIL_ID MilSystem     ;
  MIL_ID MilDigitizer  ;
  MIL_ID MilDisplay    ;
  MIL_ID MilImageDisp  ;
  MIL_ID MilGrabBufferList[BUFFERING_SIZE_MAX] = { 0 };
  long   MilGrabBufferListSize;
  long   ProcessFrameCount  = 0;
  long   NbFrames           = 0;
  double ProcessFrameRate   = 0;
  HookDataStruct UserHookData;

  /* Allocate defaults. */
  MappAllocDefault(M_SETUP, &MilApplication, &MilSystem, &MilDisplay,
                                   &MilDigitizer, &MilImageDisp);

  /* Allocate the grab buffers and clear them. */
  MappControl(M_ERROR, M_PRINT_DISABLE);
  for(MilGrabBufferListSize = 0; MilGrabBufferListSize<BUFFERING_SIZE_MAX; MilGrabBufferListSize++)
    {
    MbufAlloc2d(MilSystem,
            MdigInquire(MilDigitizer, M_SIZE_X, M_NULL),
            MdigInquire(MilDigitizer, M_SIZE_Y, M_NULL),
            M_DEF_IMAGE_TYPE,
            M_IMAGE+M_GRAB+M_PROC,
            &MilGrabBufferList[MilGrabBufferListSize]);

    if (MilGrabBufferList[MilGrabBufferListSize])
      {
      MbufClear(MilGrabBufferList[MilGrabBufferListSize], 0xFF);
      }
    else
      break;
    }
```

Multi-buffered image capture and processing (continued)

```
  MappControl(M_ERROR, M_PRINT_ENABLE);
  /* Free a buffer to leave space for possible temporary buffer. */
  MilGrabBufferListSize--;
  MbufFree(MilGrabBufferList[MilGrabBufferListSize]);

  /* Print a message. */
  printf("\nMULTIPLE BUFFERED PROCESSING.\n");
  printf("-----------------------------\n\n");
  printf("Press <Enter> to start.\n\n");

  /* Grab continuously on the display and wait for a key press. */
  MdigGrabContinuous(MilDigitizer, MilImageDisp);
  getch();

  /* Halt continuous grab. */
  MdigHalt(MilDigitizer);

  /* Initialize the User's processing function data structure. */
  UserHookData.MilImageDisp        = MilImageDisp;
  UserHookData.ProcessedImageCount = 0;

  /* Start the processing. The processing function is called for every frame grabbed. */
  MdigProcess(MilDigitizer, MilGrabBufferList, MilGrabBufferListSize,
                  M_START, M_DEFAULT, ProcessingFunction, &UserHookData);

  /* NOTE: Now the main() is free to perform other tasks while the processing is executing. */
  /* ------------------------------------------------------------------------------------- */

  /* Print a message and wait for a key press after a minimum number of frames. */
  printf("Press <Enter> to stop.\n\n");
  getch();

  /* Stop the processing. */
  MdigProcess(MilDigitizer, MilGrabBufferList, MilGrabBufferListSize,
          M_STOP, M_DEFAULT, ProcessingFunction, &UserHookData);

  /* Print statistics. */
  MdigInquire(MilDigitizer, M_PROCESS_FRAME_COUNT,  &ProcessFrameCount);
  MdigInquire(MilDigitizer, M_PROCESS_FRAME_RATE,   &ProcessFrameRate);
  printf("\n\n%ld frames grabbed at %.1f frames/sec (%.1f ms/frame).\n",
                ProcessFrameCount, ProcessFrameRate, 1000.0/ProcessFrameRate);
  printf("Press <Enter> to end.\n\n");
  getch();

  /* Free the grab buffers. */
  while(MilGrabBufferListSize > 0)
    MbufFree(MilGrabBufferList[--MilGrabBufferListSize]);

  /* Release defaults. */
  MappFreeDefault(MilApplication, MilSystem, MilDisplay, MilDigitizer, MilImageDisp);
}

/* User's processing function called every time a grab buffer is modified. */
/* --------------------------------------------------------------------------*/
```

Multi-buffered image capture and processing (continued)

```
/* Local defines. */
#define STRING_LENGTH_MAX  20
#define STRING_POS_X       20
#define STRING_POS_Y       20
long MFTYPE ProcessingFunction(long HookType, MIL_ID HookId, void MPTYPE *HookDataPtr)
  {
  HookDataStruct *UserHookDataPtr = (HookDataStruct *)HookDataPtr;
  MIL_ID ModifiedBufferId;
  MIL_TEXT_CHAR Text[STRING_LENGTH_MAX]= {'\0',};

  /* Retrieve the MIL_ID of the grabbed buffer. */
  MdigGetHookInfo(HookId, M_MODIFIED_BUFFER+M_BUFFER_ID, &ModifiedBufferId);

  /* Print and draw the frame count. */
  UserHookDataPtr->ProcessedImageCount++;
  printf("Processing frame #%d.\r", UserHookDataPtr->ProcessedImageCount);
  MOs_ltoa(UserHookDataPtr->ProcessedImageCount, Text, 10);
  MgraText(M_DEFAULT, ModifiedBufferId, STRING_POS_X, STRING_POS_Y, Text);

  /* Perform the processing and update the display. */
  #if (!M_MIL_LITE)
    MimArith(ModifiedBufferId, M_NULL, UserHookDataPtr->MilImageDisp, M_NOT);
  #else
    MbufCopy(ModifiedBufferId, UserHookDataPtr->MilImageDisp);
  #endif

  return 0;
  }
```

OCR (MIL example)

This program calibrates an OCR font (semi-font) and uses it to read a string present in the image. The string is then printed to the screen and the calibrated font is saved to disk.

"ocrsemi1.mim"

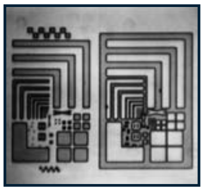
```
#include <stdio.h>
#include <string.h>
#include <mil.h>

/* Target image character specifications. */
#define CHAR_IMAGE_FILE              "ocrsemi1.mim"
#define CHAR_SIZE_X_MIN              22.0
#define CHAR_SIZE_X_MAX              23.0
#define CHAR_SIZE_X_STEP             0.50
#define CHAR_SIZE_Y_MIN              43.0
#define CHAR_SIZE_Y_MAX              44.0
#define CHAR_SIZE_Y_STEP             0.50

/* Target reading specifications. */
#define READ_REGION_POS_X            30L
#define READ_REGION_POS_Y            40L
#define READ_REGION_WIDTH            420L
#define READ_REGION_HEIGHT           70L
#define READ_SCORE_MIN               50.0

/* Font file names. */
#define FONT_FILE_IN                 "semi1292.mfo"
#define FONT_FILE_OUT                "semicali.mfo"

/* Length of the string to read (null terminated) */
#define STRING_LENGTH                13L
#define STRING_CALIBRATION           "M940902-MXD5"

/* Drawing color for the resulting string */
#define STRING_DRAWING_COLOR         255L

void main(void)
{
 MIL_ID MilApplication,              /* Application identifier. */
        MilSystem,                   /* System identifier. */
        MilDisplay,                  /* Display identifier. */
        MilImage,                    /* Image buffer identifier. */
        MilSubImage,                 /* Sub-image buffer identifier. */
        OcrFont,                     /* OCR font identifier. */
        OcrResult;                   /* OCR result buffer identifier. */
 char   String[STRING_LENGTH];       /* Array of characters to read. */
 double Score;                       /* Reading score. */

 /* Allocate defaults */
 MappAllocDefault(M_SETUP, &MilApplication, &MilSystem, &MilDisplay, M_NULL, &MilImage);

 /* Load source image into image buffer. */
 MbufLoad(CHAR_IMAGE_FILE, MilImage);

 /* Restrict the region of the image where to read the string.*/
 MbufChild2d(MilImage, READ_REGION_POS_X, READ_REGION_POS_Y, READ_REGION_WIDTH,
            READ_REGION_HEIGHT, &MilSubImage);
 /* Restore the OCR character font from disk. */
 MocrRestoreFont(FONT_FILE_IN, M_RESTORE, MilSystem, &OcrFont);

/* Pause to show the original image and ask the calibration string. */
 printf("The OCR font will be calibrated using the displayed image.\n");
 printf("\nCalibrating font...\n\n");
```

OCR (continued)

```
    /* Calibrate the OCR font. */
    MocrCalibrateFont(MilSubImage, OcrFont, STRING_CALIBRATION, CHAR_SIZE_X_MIN, CHAR_SIZE_X_MAX,
                      CHAR_SIZE_X_STEP, CHAR_SIZE_Y_MIN, CHAR_SIZE_Y_MAX, CHAR_SIZE_Y_STEP, M_DEFAULT);

    /* Set the user specific character constraints for each string position */
    MocrSetConstraint(OcrFont, 0,  M_LETTER, M_NULL);                /*  A to Z only */
    MocrSetConstraint(OcrFont, 1,  M_DIGIT,  "9");                   /*  9 only */
    MocrSetConstraint(OcrFont, 2,  M_DIGIT,  M_NULL);                /*  0 to 9 only */
    MocrSetConstraint(OcrFont, 3,  M_DIGIT,  M_NULL);                /*  0 to 9 only */
    MocrSetConstraint(OcrFont, 4,  M_DIGIT,  M_NULL);                /*  0 to 9 only */
    MocrSetConstraint(OcrFont, 5,  M_DIGIT,  M_NULL);                /*  0 to 9 only */
    MocrSetConstraint(OcrFont, 6,  M_DIGIT,  M_NULL);                /*  0 to 9 only */
    MocrSetConstraint(OcrFont, 7,  M_DEFAULT,"-");                   /*  -  only */
    MocrSetConstraint(OcrFont, 8,  M_LETTER, "M");                   /*  M only */
    MocrSetConstraint(OcrFont, 9,  M_LETTER, "X");                   /*  X only */
    MocrSetConstraint(OcrFont, 10, M_LETTER, "ABCDEFGH");            /*  SEMI checksum */
    MocrSetConstraint(OcrFont, 11, M_DIGIT,  "01234567");            /*  SEMI checksum       */

    /* Pause to signal the following read operation. */
    printf("The string present in the displayed image will be read and\n");
    printf("the result will be printed.\nPress <Enter> to continue.\n");
    getchar();

    /* Allocate an OCR result buffer. */
    MocrAllocResult(MilSystem, M_DEFAULT, &OcrResult);

    /* Read the string. */
    MocrReadString(MilSubImage, OcrFont, OcrResult);


    /* Get the string and its reading score. */
    MocrGetResult(OcrResult, M_STRING, String);
    MocrGetResult(OcrResult, M_SCORE, &Score);

    /* Print the result. */
    printf("\nThe string read is: \"%s\" (score: %.1f%%).\n\n", String, Score);

    /* Draw the string under the reading region. */
    MgraFont(M_DEFAULT, M_FONT_DEFAULT_LARGE);
    MgraColor(M_DEFAULT,STRING_DRAWING_COLOR);
    MgraText(M_DEFAULT, MilImage, READ_REGION_POS_X+(READ_REGION_WIDTH/4),
             READ_REGION_POS_Y+READ_REGION_HEIGHT, String);

    /* Save the calibrated font if the reading score was sufficient. */
    if (Score > READ_SCORE_MIN)
    {
       MocrSaveFont(FONT_FILE_OUT, M_SAVE, OcrFont);
       printf("Read successful, calibrated OCR font was saved to disk.\n");
    }

 else
    {
       printf("Error: Read score too low, calibrated OCR font not saved.\n");
    }

    printf("Press <Enter> to end.\n");
    getchar();

    /* Free all allocations. */
    MocrFree(OcrFont);
    MocrFree(OcrResult);
    MbufFree(MilSubImage);
    MappFreeDefault(MilApplication, MilSystem, MilDisplay, M_NULL, MilImage);
    }
```

Pattern matching (MIL example)

This program finds the horizontal and vertical displacement of a wafer image.



"wafer.mim"

```
#include <stdio.h>

#include <mil.h>

/* Source and target images file specifications. */
#define MODEL_IMAGE_FILE                    "wafer.mim"
#define TARGET_IMAGE_FILE                   "shfwafer.mim"
#define IMAGE_WIDTH                         512L
#define IMAGE_HEIGHT                        480L

/* Model width, height, maximum displacement, initial position */
#define MODEL_WIDTH                         64L
#define MODEL_HEIGHT                        64L

void main(void)
{

 MIL_ID MilApplication,                      /* Application identifier. */
        MilSystem,                           /* System identifier. */
        MilDisplay,                          /* Display identifier.  */
        MilImage,                            /* Image buffer identifier. */
        MilSubImage,                         /* Sub-image buffer identifier. */
        Model,                               /* Model identifier. */
        Result;                              /* Result buffer identifier. */
long    PosX, PosY;                          /* Model position. */
long    AllocError;                          /* Allocation error variable. */
double  OrgX=0.0, OrgY=0.0;                  /* Original center of model. */
double  x=0.0, y=0.0, Score=0.0;             /* Result variables. */

/* Allocate defaults. */
MappAllocDefault(M_SETUP, &MilApplication, &MilSystem, &MilDisplay, M_NULL, &MilImage);

/* Load model image into an image buffer. */
MbufLoad(MODEL_IMAGE_FILE, MilImage);

/* Restrict the region to be processed to the bottom right corner of the image. */
MbufChild2d(MilImage, IMAGE_WIDTH/2, IMAGE_HEIGHT/2, IMAGE_WIDTH/2, IMAGE_HEIGHT/2, &MilSubImage);

/* Announce the automatic model definition. */
printf("A model is being automatically defined in the source image, ");
printf("please wait...\n\n");

/* Automatically allocate normalized grayscale type model. */
MpatAllocAutoModel(MilSystem, MilSubImage, MODEL_WIDTH, MODEL_HEIGHT, M_DEFAULT,
                   M_DEFAULT, M_NORMALIZED, M_DEFAULT, &Model);

/* Check for a successful model allocation. */
MappGetError(M_CURRENT, &AllocError);
if (!AllocError)
  {
  MpatInquire(Model,M_ALLOC_OFFSET_X+M_TYPE_LONG,&PosX);
  MpatInquire(Model,M_ALLOC_OFFSET_Y+M_TYPE_LONG,&PosY);
  MpatInquire(Model,M_ORIGINAL_X,&OrgX);
  MpatInquire(Model,M_ORIGINAL_Y,&OrgY);
```

Pattern matching (continued)

```
   /* Draw box around model. */
   MgraRect(M_DEFAULT, MilSubImage, PosX - 1, PosY - 1, PosX + MODEL_WIDTH, PosY + MODEL_HEIGHT);
   printf("Model successfully defined as shown on the displayed image.\n");
   printf("Press <Enter> to continue.\n");
   getchar();

   /* Load target image into an image buffer. */
   MbufLoad(TARGET_IMAGE_FILE, MilImage);

   /* Allocate result. */
   MpatAllocResult(MilSystem, 1L, &Result);

   /* Find model. */
   MpatFindModel(MilSubImage, Model, Result);

   /* If one model was found above the acceptance threshold set. */
   if (MpatGetNumber(Result, M_NULL) == 1L)
     {
     /* Get results. */
     MpatGetResult(Result, M_POSITION_X, &x);
     MpatGetResult(Result, M_POSITION_Y, &y);
     MpatGetResult(Result, M_SCORE, &Score);

     /* Draw a box around occurrence. */
     MgraRect(M_DEFAULT, MilSubImage,
           (long)(x + 0.5) - (MODEL_WIDTH/2)   - 1,
           (long)(y + 0.5) - (MODEL_HEIGHT/2) - 1,
           (long)(x + 0.5) + (MODEL_WIDTH/2),
           (long)(y + 0.5) + (MODEL_HEIGHT/2));

     /* Analyze and print results. */
     printf("A misaligned version of the source image was loaded.\n\n");
     printf("Image was found to be offset by %.2f in X, and %.2f in Y.\n", x - OrgX, y - OrgY);
     printf("Model match score is %.1f percent.\n", Score);
     printf("Press <Enter> to end.\n");
     getchar();
     }
   else
    {
    printf("Error: Pattern not found properly.\n");
    printf("Press <Enter> to end.\n");
    getchar();
    }

   /* Free result buffer and model. */
   MpatFree(Result);
   MpatFree(Model);
   }
    else
   {
   printf("Error: Automatic model definition failed.\n");
   printf("Press <Enter> to end.\n");
   getchar();
   }

/* Free child image and defaults. */
MbufFree(MilSubImage);
MappFreeDefault(MilApplication, MilSystem, MilDisplay, M_NULL, MilImage);
}
```

String Reader (MIL example)

This program uses the String Reader module to define a font from an image containing a mosaic of license plates. Two string models are then defined and parameterized to read only valid license plates. License plate reading is then performed in a target image of a car on a road.

```
#include <mil.h>
#include <conio.h>

/* MIL image file specifications. */
#define IMAGE_FILE_DEFINITION M_IMAGE_PATH MIL_TEXT("QcPlates.mim")
#define IMAGE_FILE_TO_READ    M_IMAGE_PATH MIL_TEXT("LicPlate.mim")

/* String containing all characters used for font definition. */
#define TEXT_DEFINITION       "AVS300CVK781JNK278 EBX380QKN918HCC839 YRH900ZQR756977AMQ GPK742389EYE569ESQ"

/* Font normalization size Y. */
#define NORMALIZATION_SIZE_Y    20L

/* Max size of plate string. */
#define STRING_MAX_SIZE         32L


/*****************************************************************************/
/* Main. */
void main(void)
  {
  MIL_ID MilApplication,                 /* Application identifier.      */
         MilSystem,                      /* System identifier.           */
         MilDisplay,                     /* Display identifier.          */
         MilImage,                       /* Image buffer identifier.     */
         MilOverlayImage,                /* Overlay image.               */
         MilStrContext,                  /* String context identifier.   */
         MilStrResult;                   /* String result buffer identifier. */
  long   NumberOfStringRead;             /* Total number of strings to read. */
  double Score;                          /* String score.                */
  char*  StringResult[STRING_MAX_SIZE+1]; /* String of characters read.  */
  double Time = 0.0;                     /* Time variable.               */

  /* Print module name. */
  printf("\nSTRING READER MODULE:\n");
  printf("---------------------\n\n");

  /* Allocate defaults */
  MappAllocDefault(M_SETUP, &MilApplication, &MilSystem, &MilDisplay, M_NULL, M_NULL);

  /* Restore the font definition image */
  MbufRestore(IMAGE_FILE_DEFINITION, MilSystem, &MilImage);

  /* Display the image and prepare for overlay annotations. */
  MdispSelect(MilDisplay, MilImage);
  MdispControl(MilDisplay, M_OVERLAY, M_ENABLE);
  MdispInquire(MilDisplay, M_OVERLAY_ID, &MilOverlayImage);

  /* Allocate a new empty String Reader context. */
  MstrAlloc( MilSystem, M_FEATURE_BASED, M_DEFAULT, &MilStrContext);

  /* Allocate a new empty String Reader result buffer. */
  MstrAllocResult(MilSystem, M_DEFAULT, &MilStrResult);

  /* Add a new empty user defined font to the context. */
  MstrControl(MilStrContext, M_CONTEXT, M_FONT_ADD, M_USER_DEFINED);
```

“QcPlates.mim”

“LicPlate.mim”

String Reader (continued)

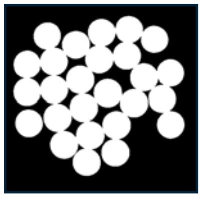
```
/* Add user defined characters from the license plate mosaic image. */
MstrEditFont(MilStrContext, M_FONT_INDEX(0), M_CHAR_ADD,
            M_USER_DEFINED + M_FOREGROUND_BLACK,
            MilImage, TEXT_DEFINITION, M_NULL);

/* Draw all the characters in the font in the overlay image. */
MgraColor(M_DEFAULT, M_COLOR_GREEN);
MstrDraw(M_DEFAULT, MilStrContext, MilOverlayImage, M_DRAW_CHAR,
        M_FONT_INDEX(0), M_NULL, M_ORIGINAL);

/* Normalize the characters of the font to an appropriate size. */
MstrEditFont(MilStrContext, M_FONT_INDEX(0), M_CHAR_NORMALIZE,
            M_SIZE_Y, NORMALIZATION_SIZE_Y, M_NULL, M_NULL);

/* Add 2 new empty strings models to the context for the 2 valid types of
   plates (3 letters followed by 3 numbers or 3 numbers followed by 3 letters)
   Note that the read time increases with the number of strings in the context.
*/
MstrControl(MilStrContext, M_CONTEXT, M_STRING_ADD, M_USER_DEFINED);
MstrControl(MilStrContext, M_CONTEXT, M_STRING_ADD, M_USER_DEFINED);

/* Set number of strings to read. */
MstrControl(MilStrContext, M_CONTEXT, M_STRING_NUMBER, 1);

/* Set number of expected characters for all string models to 6 characters. */
MstrControl(MilStrContext, M_STRING_INDEX(M_ALL), M_STRING_SIZE_MIN, 6);
MstrControl(MilStrContext, M_STRING_INDEX(M_ALL), M_STRING_SIZE_MAX, 6);

/* Set default constraint to uppercase letter for all string models */
MstrSetConstraint(MilStrContext, M_STRING_INDEX(0), M_DEFAULT, M_LETTER + M_UPPERCASE, M_NULL );
MstrSetConstraint(MilStrContext, M_STRING_INDEX(1), M_DEFAULT, M_LETTER + M_UPPERCASE, M_NULL );

/* Set constraint of 3 last characters to digit for the first string model */
MstrSetConstraint(MilStrContext, M_STRING_INDEX(0), 3, M_DIGIT, M_NULL );
MstrSetConstraint(MilStrContext, M_STRING_INDEX(0), 4, M_DIGIT, M_NULL );
MstrSetConstraint(MilStrContext, M_STRING_INDEX(0), 5, M_DIGIT, M_NULL );

/* Set constraint of 3 first characters to digit for the second string model */
MstrSetConstraint(MilStrContext, M_STRING_INDEX(1), 0, M_DIGIT, M_NULL );
MstrSetConstraint(MilStrContext, M_STRING_INDEX(1), 1, M_DIGIT, M_NULL );
MstrSetConstraint(MilStrContext, M_STRING_INDEX(1), 2, M_DIGIT, M_NULL );

/* Pause to show the font definition. */
printf("This program has defined a font with this Quebec plates mosaic image.\n");
printf("Press <Enter> to continue.\n\n");
getch();

/* Clear the display overlay. */
MdispControl(MilDisplay, M_OVERLAY_CLEAR, M_DEFAULT);

/* Load image to read into image buffer. */
MbufLoad(IMAGE_FILE_TO_READ, MilImage);

/* Preprocess the String Reader context. */
MstrPreprocess(MilStrContext, M_DEFAULT);

/* First, perform a dummy read for better function timing accuracy (model cache effet,...). */
MstrRead(MilStrContext, MilImage, MilStrResult);

/* Reset the timer. */
MappTimer(M_TIMER_RESET+M_SYNCHRONOUS, M_NULL);
```

String Reader (continued)

```
/* Perform the read operation on the specified target image. */
MstrRead(MilStrContext, MilImage, MilStrResult);

/* Read the time. */
MappTimer(M_TIMER_READ+M_SYNCHRONOUS, &Time);

/* Get number of strings read and show the result. */
MstrGetResult(MilStrResult, M_GENERAL, M_STRING_NUMBER + M_TYPE_LONG, &NumberOfStringRead);
if( NumberOfStringRead >= 1)
  {
  printf("The license plate was read successfully (%.2lf ms)\n\n", Time*1000 );

  /* Draw read result. */
  MgraColor(M_DEFAULT, M_COLOR_BLUE);
  MstrDraw(M_DEFAULT, MilStrResult, MilOverlayImage, M_DRAW_STRING, M_ALL, M_NULL, M_DEFAULT);
  MgraColor(M_DEFAULT, M_COLOR_GREEN);
  MstrDraw(M_DEFAULT, MilStrResult, MilOverlayImage, M_DRAW_STRING_BOX, M_ALL, M_NULL, M_DEFAULT);

  /* Print the read result. */
  printf(" String                 Score\n" );
  printf(" ----------------------------------------\n" );
  MstrGetResult(MilStrResult, 0, M_STRING, StringResult);
  MstrGetResult(MilStrResult, 0, M_STRING_SCORE, &Score);
  printf(" %s                 %.1lf\n", StringResult, Score );
  }
else
  {
  printf("Error: Plate was not read.\n");
  }

/* Pause to show results. */
printf("\nPress <Enter> to end.\n\n");
getch();

/* Free all allocations. */
MstrFree(MilStrContext);
MstrFree(MilStrResult);
MbufFree(MilImage);

/* Free defaults. */
MappFreeDefault(MilApplication, MilSystem, MilDisplay, M_NULL, M_NULL);
}
```

Watershed segmentation (MIL example)

This program uses watershed and distance functions to separate touching objects in a binary image.



"binpills.mim"

```
/* Regular includes. */
#include <mil.h>
#include <stdio.h>

/* Source image specifications. */
#define IMAGE_FILE        "binpills.mim"

/* Minimal distance variations for the watershed calculation. */
#define WSHED_MINIMAL_DISTANCE_VARIATION       2

/* Main application function */
void main()
{
  MIL_ID  MilApplication,              /* Application identifier. */
          MilSystem,                   /* System identifier. */
          MilDisplay,                  /* Display identifier. */
          MilImage,                    /* Image buffer identifier. */
           MilImageWatershed;          /* Image buffer identifier. */

  /* Allocate defaults. */
  MappAllocDefault(M_SETUP, &MilApplication, &MilSystem, &MilDisplay, M_NULL, MilImage);

  /* Restore the source image into an automatically allocated
  * image work buffer and copy it to the display image.
  */
  MbufRestore(IMAGE_FILE, MilSystem, &MilImageWatershed);
  MbufCopy(MilImageWathershed, &MilImage);

  /* Pause to show the original image. */
  printf("Original image.\n");
  printf("Press <Enter> to continue.\n\n");
  getchar();

  /* Perform a distance transformation on the binary image. */
  MimDistance(MilImage, MilImageWatershed, M_CHAMFER_3_4);

  /* Find the watersheds of the distance image. */
  MimWatershed(MilImageWatershed, M_NULL, MilImageWatershed, WSHED_MINIMAL_DISTANCE_VARIATION,
                  M_STRAIGHT_WATERSHED + M_MAXIMA_FILL + M_SKIP_LAST_LEVEL);

  /* AND the watershed image with the original binary image
   * to separate the touching pills.
   */
  MimArith(MilImageWatershed, MilImage, MilImage, M_AND);

  /* Pause to show the segmented image. */
  printf("Segmented image.\n");
  printf("Press <Enter> to end.\n\n");
  getchar();

  /* Free all allocations */
  MbufFree(MilImageWatershed);
  MappFreeDefault(MilApplication, MilSystem, MilDisplay, M_NULL, MilImage);
}
```

ActiveMIL Control Listing and Description

This section provides an overview of each ActiveMIL control and a brief description of most ActiveMIL command control-actions (methods), control-generated episodes (events) and control-states (properties). Note that this represents only a partial list of available commands. For a complete description of the syntax and the use of each command, refer to the ActiveMIL On-line Control Reference manual.

Application control

Used to initialize and control the ActiveMIL application environment. The Application control includes control of integrated debugging features, system resource compensation, command threads and related events, as well as a timer function. On-board thread use is also controlled by the Application control.

Properties	Description
AvailableSystems	Returns the collection of systems that are available for use in a PC and are accessible to the Application control.
MemoryCompensation	Returns or sets whether on-board memory compensation on the Host is enabled.
NonPagedMemory	Allows you to determine the non-paged memory (DMA) settings of the application.
ProcessingCompensation	Returns or sets whether on-board processing compensation on the Host is enabled.
ResultsValidation	Returns or sets whether results validation is enabled when using an analysis control's Results.Get method.
Timer	Allows you to manipulate the high-resolution timer of the application.

System control

Used to control the ActiveMIL system (frame grabber boards, vision processor boards, or host system).

Events	Description
SerialPortData	Occurs when data is received on a serial port.
System.ControlUserBits	Controls the specified setting for the specified user-defined signal.
System.InquireUserBits	Returns the value of a specified setting for a specified user-defined signal.
UserBitChanged	Occurs when the state of a user bit changes in accordance with its specified interrupt mode (edge rising or falling).

Methods	Description
ControlUserBits	Controls the specified setting for the specified user-defined signal.
InquireUserBits	Returns the value of a specified setting for a specified user-defined signal.
ShowPropertyPages	Opens the specified property pages of the System control in a window.

Properties	Description
DeviceNumber	Returns or sets the device number (or rank) of the System control.
GrabInDisplay	Allows you to specify or determine the interaction between the digitizer and the display during a displayed grab operation.
NumberofCRTControllers	Returns the number of CRT controllers available in a system.
NumberOfDigitizers	Returns the number of digitizers available in a system.
NumberOfProcessors	Returns the number of processors available in a system.
SerialPorts	Returns the collection of serial ports available to the System control, allowing access to each of its elements.
SystemDescriptor	Returns or sets the descriptor of the system to allocate.
SystemType	Returns or sets the type of system.
UserBits	Allows you to specify or determine the auxiliary I/O pins on the Matrox 4Sight/4Sight-II platform.
Watchdog	Allows you to specify or determine the settings of the Watchdog.

BlobAnalysis control

Used to identify and measure connected components (blobs) in an image.

Methods	Description
ApplyFilter	Applies a filter that specifies the blobs to include, exclude or delete permanently from the blob results.
Calculate	Identifies the blobs in an image and calculates selected features.
EraseBorderBlobs	Copies all blobs that do not touch the borders of the source image into the destination image.
ExtractHoles	Copies all holes within blobs from the source image to the destination image.
FillHoles	Copies all blobs from the source image into the destination image and fills blob holes.
LabelImage	Labels each included blob with its own unique label.
Reconstruct	Copies all blobs, from the source image, that have at least one corresponding non-zero seed pixel in the seed image to the destination image.
Results.Draw	Draw specific features of the blob analysis results in the destination image.
ShowPropertyPages	Opens the specified property pages of the BlobAnalysis control in a window.

Properties	Description
FeatureList	Allows you to select features for calculation (see list below).
Filters	A collection that allows blob results to be filtered in or out of the results collection depending on calculated feature values.
Results	Returns the collection of results calculated for the BlobAnalysis control, allowing access to its properties. (See list)
SortingKeys	A collection of sorting keys that allow blob results to be sorted depending on a calculated feature values.

Events	Description
ResultsModified	Occurs after results have been modified.

Blob features and results

For the BlobAnalysis control, the result(s) and feature(s) that can be calculated include(s):

Features

- Area
- Box
- Breadth
- CenterofGravity
- CentralMovementX0Y2
- CentralMovementX1Y1
- CentralMovementX2Y0
- Chains
- Compactness
- ContactPoints
- ConvexPerimeter
- Elongation
- EulerNumber
- FeretElongation
- FirstPointX
- FirstPointY
- GeneralFeret
- GeneralMoment
- Intercept0
- Intercept45
- Intercept90
- Intercept135
- Length
- MaximumFeretAngle
- MaximumFeretDiameter
- MaximumPixelValue
- MeanFeretDiameter
- MeanPixelValue
- MinimumFeretAngle
- MinimumFeretDiameter
- MinimumPixelValue
- MomentX2Y0
- MomentX1Y1
- MomentX1Y0
- MomentX0Y2
- MomentX0Y1
- NumberofHoles
- Perimeter
- PrincipleAxisAngle
- Roughness
- Runs
- SecondaryAxisAngle
- SigmaOfPixelValues
- SumOfPixelValues
- SumOfSquaredPixelValues

Results

- Area
- BoxXMaximum
- BoxXMinimum
- BoxYMaximum
- BoxYMinimum
- Breadth
- CenterOfGravityX
- CenterOfGravityXGrayscale
- CenterOfGravityY
- CenterOfGravityYGrayscale
- CentralMomentX0Y2
- CentralMomentX0Y2Grayscale
- CentralMomentX1Y1
- CentralMomentX1Y1Grayscale
- CentralMomentX2Y0
- CentralMomentX2Y0Grayscale
- Chains
- Compactness
- ContactPointXMaximumAtYMaximum
- ContactPointXMinimumAtYMinimum
- ContactPointYMaximumAtXMaximum
- ContactPointYMinimumAtXMinimum
- ConvexPerimeter
- Elongation
- EulerNumber
- FeretElongation
- FeretX
- FeretY
- FirstPointX
- FirstPointY
- GeneralFeret
- GeneralMoment
- GeneralMomentGrayscale
- Intercept0
- Intercept45
- Intercept90
- Intercept135
- LabelValue
- Length
- MaximumFeretAngle
- MaximumFeretDiameter
- MaximumPixelValue
- MinimumFeretDiameter
- MeanFeretDiameter
- MeanPixelValue
- MinimumFeretAngle
- MinimumFeretDiameter
- MinimumPixelValue
- MomentX0Y1
- MomentX0Y1Grayscale
- MomentX0Y2
- MomentX0Y2Grayscale
- MomentX1Y0
- MomentX1Y0Grayscale
- MomentX1Y1
- MomentX1Y1Grayscale
- MomentX2Y0
- MomentX2Y0Grayscale
- NumberofHoles
- Perimeter
- PrincipleAxisAngle
- PrincipleAxisAngleGrayscale
- Roughness
- Runs
- SecondaryAxisAngle
- SecondaryAxisAngleGrayscale
- SigmaOfPixelValues
- SumOfPixelValues
- SumOfSquaredPixelValues

Calibration control

Used to convert coordinates or measurements from pixel to real-world units, as well as to correct distortions in an image.

Methods	Description
CalibrateGrid	Calibrates your imaging setup using a grid.
CalibratePoints	Calibrates your imaging setup using a list of coordinates.
ConvertCoordinatePixelToWorld	Converts a pair of coordinates from their pixel value to their world value.
ConvertCoordinateWorldToPixel	Converts a pair of coordinates from their world value to their pixel value.
ConvertResultPixelToWorld	Converts a non-positional result (length, area, or angle) from its pixel value to its world value.
ConvertResultWorldToPixel	Converts a non-positional result (length, area, or angle) from its world value to its pixel value.
CorrectImage	Physically transforms an image to remove certain types of distortions.
Load	Loads the calibration object from a file, into the Calibration control.
LoadStream	Loads the settings of a previously saved Calibration control from a file or memory.
Save	Saves the calibration object in a file.
SaveStream	Saves the settings of a Calibration control to a specified file or memory.
ShowPropertyPages	Opens the specified property pages of the Calibration control in a window.

Properties	Description
CalibrationPoints	Returns the collection of calibration points of the Calibration control, allowing access to the collection's elements.
CalibrationMode	Returns or sets the calibration mode (i.e. Linear Interpolation, Perspective Transformation).
CameraPosition	Allows you to specify or determine the position of the camera relative to the absolute coordinate system.
Grid	Allows you to specify or determine the calibration grid attributes.
OutputCoordinateSystem	Returns or sets the output coordinate system in which to return results from operations on calibrated images.
RelativeOrigin	Allows you to specify or determine the origin and/or orientation of the relative coordinate system.
Results	Allows you to obtain pixel characteristics results from a Calibrated control.
TransformCacheEnabled	Returns or sets whether to enable or disable a cache to accelerate the CorrectImage method.

Events	Description
ResultsModified	Occurs after results have been modified.

CharacterRecognition control

Template-based character recognition control that includes character font definition, as well as font archiving and retrieving.

Methods	Description
CalibrateFont	Calibrates the font's character size to match the dimensions of the specified sample image.
ConvertOCRType	Converts the search algorithm used by the CharacterRecognition control from one type to the other.
Load	Loads the font from a file, into the CharacterRecognition control.
Preprocess	Preprocesses the CharacterRecognition control.
ReadString	Reads a string from the target image.
Save	Saves the font data in a file.
ShowPropertyPages	Opens the specified property pages of the Character Recognition control in a window.
VerifyString	Verifies that a known string is present in the image.

Properties	Description
AcceptanceThreshold	Returns or sets the minimum acceptance level for an entire string to be accepted during a read or verify operation.
Constraints	Returns the collection of constraints available to the CharacterRecognition control, allowing access to its elements.
ContrastEnhancement	Returns or sets whether the contrast enhancement step is performed during read or verify operations.
EnableBlankCharacters	Returns or sets whether blank characters should be found in the string.
EnableBrokenCharacters	Returns or sets whether broken characters can be read/verified.
EnableTouchingCharacters	Returns or sets whether touching characters can be read/verified.
Font	Allows you to specify or determine the characteristics of the font.
ForegroundPixelValue	Returns or sets whether the characters are brighter or darker than the background.
MaximumStringLength	Returns or sets the maximum length of the string to be read or verified.
NumberOfStrings	Returns or sets the number of strings to be found in the target image.
Results	Allows you to obtain the results of the last read or verify operation.
SearchAngle	Allows you to specify or determine the CharacterRecognition control's angular search range.
SearchRegion	Allows you to specify or determine the region of interest in which to search for the string.
Speed	Returns or sets the speed factor.
StringLocation	Returns or sets whether the string location step is performed during read or verify operations.
StringType	Returns or sets the type of font being defined.
TargetCharacter	Allows you to specify or determine the characteristics of the font characters.

Events	Description
ResultsModified	Occurs after results have been modified.
ValidateString	Occurs after a read or verify operation.

Code control

Used to read (and write) various 1D and 2D code symbologies.

Methods	Description
CalculateCodeSize	Get the minimum X and Y size required for the destination image of a write operation.
Load	Loads the code from a file into the Code control.
LoadStream	Loads the settings of a previously saved Code control from a file or memory.
ReadCode	Read the specified type of code in an image.
Save	Saves the code data into a file.
SaveStream	Saves the settings of a Code control to a specified file or memory.
ShowPropertyPages	Opens the specified property pages of the Code control in a window.
VerifyCode	Computes the different quality grades of the code included in the specified source image.
WriteCode	Encode an ASCII string into an image.

Properties	Description
CodeType	Returns or sets the type of code to read or write (see list below).
EncodingType	Returns or sets the encoding type (see list below).
ErrorCorrectionType	Returns or sets the type of error correction (see list below).
Results	Allows you to obtain results from the last read or write operation.
SearchRegion	Allows you to specify the region of the image in which to search for the code.
TargetGrid	Allows you to specify read or write boundaries, in order to improve robustness.

Events	Description
ResultsModified	Occurs after results have been modified.

1D and 2D code symbologies

For the respective methods, the code type(s) that can be read or written include(s):

Code Types	Encoding Types	Error Correction
BC412	Standard encoding type	No error correction
Codabar	Standard encoding type	No error correction
Code39	ASCII encoding, Standard encoding type	No error correction; check-digit error correction
Code93	ASCII encoding	Check-digit error correction
Code128 (UCC/EAN128)	ASCII encoding	Check-digit error correction
DataMatrix	Numeric encoding, Alpha encoding, AlphaNumericPunc encoding, AlphaNumeric encoding, ASCII encoding, IS08 encoding	10, 40, 50, 60, 70, 80,90, 100, 110, 120, 130, 140 or 200 error correction
EAN8	Numeric encoding	Check-digit error correction

Continued...

Code control (continued)

1D and 2D code symbologies

For the respective methods, the code type(s) that can be read or written include(s):

Code Types	Encoding Types	Error Correction
EAN13	Numeric encoding	Check-digit error correction
Interleaved 2/5	Numeric encoding	No error correction; check-digit error correction
Maxicode	Encoding mode 2, 3, 4, 5, 6	Reed Solomon error correction
MicroPDF417	Standard encoding type	Reed Solomon error correction
PDF417 (2D)	Standard encoding type	Reed Solomon 1 - 8 error correction
Pharma code	Numeric encoding	No error correction
Planet code	Numeric encoding	Check-digit error correction
Postnet code	Numeric encoding	Check-digit error correction
QR	QR code Model 1, 2 encoding	Lowest-level QR, Low-level QR, High-level QR, Highest-level QR
RSS	RSS 14, RSS 14 Stacked, RSS 14 Stacked Omni, RSS 14 Truncated, RSS Expanded RSS Expanded Stacked, RSS Limited encoding.	Check-digit error correction
UPC-A	Numeric encoding	Check-digit error correction
UPC-E	Numeric encoding	Check-digit error correction

Composite code symbologies

This code type is a composite of a 1D (RSS, UPC-A, UPC-E, EAN-8, EAN-13, or UCC/EAN128) and a 2D code type (PDF417 or MicroPDF417).

Digitizer control

Used to initialize and control a digitizer (image capture device), which includes capture mode (trigger, frame/field, blocking/non-blocking), image scaling and cropping, input channel, input LUT, analog settings (references, hue, saturation, and brightness) as well as events for callback functions.

Methods	Description
Focus	Adjusts the camera's lens motor to a position which provides optimum focus.
Grab	Grabs data from an input device into an image.
GrabContinuous	Grabs data continuously from an input device.
GrabWait	Waits for the end of the grab in progress.
Halt	Halts a continuous grab from an input device.
SendTrigger	Sends a software trigger to the specified digitizer if its trigger source is set to software.
ShowPropertyPages	Opens the specified property pages of the Digitizer control in a window.

Properties	Description
BlackReference	Returns or sets the digitization black reference level.
Brightness	Returns or sets the digitization brightness reference level for composite input video signals.
Bayer	Allows you to specify or determine the digitizer's Bayer properties.
Calibration	Returns or sets the Calibration control to associate with the digitizer.
Channel	Returns or sets the active input channel of the digitizer.
Contrast	Returns or sets the contrast reference level of a composite input video signal.
Exposure1(2)Format	Returns or sets the state of TTL or RS-422/LVDS transmitters for the exposure signal generated by timer1 or 2.
Exposure1(2)Mode	Returns or sets the active level of the exposure signal generated by timer1 or 2.
Exposure1(2)Source	Returns or sets the trigger source for timer1 or 2.
Exposure1(2)Time	Returns or sets the grab exposure time, or the active time of timer1 or 2.
Exposure1(2)TimeDelay	Returns or sets the delay between the trigger event and the start of exposure, or sets the inactive time of timer1 or 2.
Format	Returns or sets the digitizer configuration format of the specified digitizer.
InputRegion	Allows you to specify or determine the digitizer's input region.
LUT	Allows you to specify or determine the digitizer's custom LUT.
MultipleBuffering	Allows you to build a multiple buffering application.
SerialPort	Allows you to specify or determine the digitizer's serial port properties.
SynchronizeOnGrabEnd	Returns or sets a second digitizer with which to synchronize the digitizer, so that the two digitizers grab consecutively.
TriggerEnabled	Returns or sets whether a grab should wait for a trigger.
TriggerMode	Returns or sets the hardware grab trigger activation mode.
TriggerSource	Returns or sets the signal source of the grab trigger.
UserBits	Allows you to specify or determine the state of the user-defined signals.
WhiteReference	Returns or sets the digitization white reference level.

Digitizer control (continued)

Events	Description
CameraPresent	Occurs at the start of the incoming signal's fields.
FieldStart	Occurs at the start of the incoming signal's fields.
FieldStartEven	Occurs at the start of the incoming signal's even fields.
FieldStartOdd	Occurs at the start of the incoming signal's odd fields.
FrameStart	Occurs at the start of each grabbed or displayed frame.
GrabEnd	Occurs at the end of grab.
GrabFieldEnd	Occurs at the end of each grabbed field.
GrabFieldEndEven	Occurs at the end of each grabbed even field.
GrabFieldEndOdd	Occurs at the end of each grabbed odd field.
GrabFrameEnd	Occurs at the end of each grabbed frame.
GrabFrameStart	Occurs at the start of each grabbed frame.
GrabLine	Occurs when the specified line number is reached.
GrabLineEnd	Occurs when the data of the specified line number is in the buffer and ready to be processed.
GrabStart	Occurs at the start of grab.
MoveLens	Occurs after each new focus position determined by the Focus method.
ProcessModifiedImage	Occurs when an image, in the array of Image controls passed to the MultipleBuffering.Process method, is modified.
SerialPortData	Occurs when data is received on the digitizer’s serial port.
UserBitChanged	Occurs when the user-defined signal generates an interrupt upon a rising edge.

Display control

Used to initialize and control an image display, which includes image display windows, graphics overlay, output LUT, image pan, scroll, and zoom.

Methods	Description
ClearOverlay	Returns or sets the value to which the overlay image associated with the display should be cleared.
Pan	Pans and scrolls the specified display.
ShowPropertyPages	Opens the specified property pages of the Display control in a window.
Zoom	Magnifies or reduces the view of the image being displayed.

Properties	Description
CenterDisplay	Returns or sets whether the image selected to the display will be centered in the display.
ExternalWindow	The external window is a display window, created and managed by ActiveMIL, that allows panning and zooming via scrollbars and buttons.
FillDisplay	Returns or sets whether the display will be filled with the selected image using an automatically calculated zoom factor.
LUT	Allows you to specify or determine the display's custom LUT.
OverlayImage	Allows you access to the overlay image associated with the display.
OverlayKeyColor	Returns or sets the keying color for overlay.

Events	Description
Click	Occurs when the user clicks any mouse button.
DblClick	Occurs when the user double-clicks the left mouse button.
FrameStart	Occurs at the start of each displayed frame.
KeyDown	Occurs when the user presses a key.
KeyPress	Occurs when the user presses and releases an ANSI key.
KeyUp	Occurs when the user releases a key.
MouseDown	Occurs when the user clicks a mouse button.
MouseMove	Occurs when the user moves the mouse.
MouseUp	Occurs when the user releases a mouse button.
Paint	When a section of the display object needs repainting.
ScrollHorizontal	Occurs when the content of the display changes in a manner that requires the horizontal scroll bar values to be adjusted.
ScrollVertical	Occurs when the content of the display changes in a manner that requires the vertical scroll bar values to be adjusted.
Zoom	Occurs after the display window has been zoomed.

EdgeFinder control

Used to extract and analyze object contours or thin curvilinear features.

Methods	Description
ApplyFilter	Applies a filter that specifies the edges to include, exclude or delete permanently from the edge results.
Calculate	Extracts the edges from an image and calculates selected features.
CalculateFromResult	Calculates selected features from the result of each edge found to accelerate the search time.
Draw	Draws specified edge feature in the destination image.
Load	Loads a previously saved EdgeFinder control from a file.
LoadStream	Loads the settings of a previously saved EdgeFinder control from a file or memory.
Mask	Masks the source image or edge results using the specified mask image.
PutEdgeResults	Puts edge chains from user-supplied arrays into the results of an EdgeFinder control.
Results.CalculateStat	Calculates the statistics of a characteristic of an edge or of all edges in the results of an EdgeFinder control.
Results.Draw	Draws the specified results, calculated for the edge in the destination image.
Save	Saves the settings of the EdgeFinder control to disk.
SaveStream	Saves the settings of a EdgeFinder control to a specified file or memory.
ShowPropertyPages	Opens the specified property pages of the EdgeFinder control in a window.

Properties	Description
Accuracy	Returns or sets the edgel accuracy required for the edge extraction.
DrawingParameters	Allows you to specify or determine the characteristics of the drawing parameters, such as scale factor to be used when drawing edgels with sub-pixel accuracy.
ExtractionFilter	Allows you to specify or determine the filter settings used to perform the edge extraction.
FeatureList	Allows you to select features for calculation (see list).
FillGapParameters	Allows you to specify or determine the characteristics of the gap filling parameters.
Filters	A collection that allows EdgeFinder results to be filtered in or out of the results collection depending on calculated feature values.
ImageFeature	Allows you to specify or determine which image feature(s) are selected for saving.
ModelFinderCompatible	Returns or sets whether the EdgeFinder control can be used with a Model Finder control as a target to be searched or as a model.
NearestNeighbors	Returns the collection of nearest neighbors available to the EdgelFinder control, allowing access to its elements.
Results	Returns the collection of results calculated for the EdgeFinder control, allowing access to its properties (see list).
SortingKeys	A collection of sorting keys that allow EdgeFinder results to be sorted depending on a calculated feature values.
SourceDerivativeImages	Allows you to specify or determine the derivative images containing the edges to extract.
Threshold	Allows you to specify or determine the threshold settings used to perform the edge extraction.

Events	Description
ResultsModified	Occurs after results have been modified.

EdgeFinder features and results

For the EdgeFinder control, the feature(s) and result(s) that can be calculated include(s):

Features

AverageStrength,
Box,
CenterOfGravity,
CircleFit,
Closure,
ContactPoints,
ConvexPerimeter,
EdgelAngle,
EdgelMagnitude,
ElipseFit,
FastLength,
FeretElongation,
FirstPoint,
GeneralFeret,
LabelValue,
Length,
LineFit,
MaximumFeretAngle,
MaximumFeretDiameter,
MeanFeretDiameter,
MinimumFeretAngle,
MinimumFeretDiameter,
MomentElongation,
MomentElongationAngle,
Position,
Size,
Strength,
Tortuosity

Results

AngleImage,
AverageStrength,
BoxXMaximum,
BoxXMinimum,
BoxYMaximum,
BoxYMinimum,
CenterOfGravityX,
CenterOfGravityY,
CircleFitCenterX,
CircleFitCenterY,
CircleFitCoverage,
CircleFitError,
CircleFitRadius,
Closure,
ContactPointXMaximumAtYMaximum,
ContactPointXMinimumAtYMinimum,
ContactPointYMaximumAtXMinimum,
ContactPointYMinimumAtXMaximum,
ConvexPerimeter,
CrossDerivativeImage,
Edgels,
EllipseFitCenterX,
EllipseFitCenterY,
EllipseFitCoverage,
EllipseFitError,
EllipseFitMajorAxis,
EllipseFitMinorAxis,
FastLength,
FeretElongation,
FeretX,
FeretY,
FirstDerivativeXImage,
FirstDerivativeYImage,
FirstPointX,
FirstPointY,
GeneralFeret,
GeneralFeretFirstEdgelIndex,
GeneralFeretSecondEdgelIndex,
LabelValue,
Length,
LineFitA,
LineFitB,
LineFitC,
LineFitError,
MagnitudeImage,
MaximumFeretAngle,
MaximumFeretDiameter,
MaximumFeretFirstEdgelIndex,
MaximumFeretSecondEdgelIndex,
MeanFeretDiameter,
MinimumFeretAngle,
MinimumFeretDiameter,
MinimumFeretFirstEdgelIndex,
MinimumFeretSecondEdgelIndex,
MomentElongation,
MomentElongationAngle,
PositionX,
PositionY,
SecondDerivativeXImage,
SecondDerivativeYImage,
Size,
Strength,
ThresholdHighValue,
ThresholdLowValue,
TotalNumberOfEdgels,
TotalNumberOfVertices
Tortuosity,
Vertices

GraphicContext control

Used to create drawings and text annotations in an image. This control provides a set of graphics primitives (arc, circle, line, and rectangle), control of color (foreground, background, fill) and text (font, color, size).

Methods	Description
Arc	Draws an arc.
Cross	Draws a cross, at any angle.
Dots	Draw one or more single-pixel dots.
Fill	Performs a boundary-type seed fill.
LineSegments	Draw one or more line segments.
Rectangle	Draws a rectangle, at any angle.
ShowPropertyPages	Opens the specified property pages of the Graphic Context control in a window.
Text	Writes text.

Properties	Description
BackgroundColor	Returns or sets the background color of the GraphicContext control.
DrawingRegion	Allows you to specify or determine the drawing region within the image.
Font	Returns or sets the type of font with which to write text.
ForegroundColor	Returns or sets the foreground color of the GraphicContext control.

Image control

Used to allocate and control ActiveMIL images, and to generate data for a LUT and the warp function. Includes control of a child buffer (ROI), image compression and decompression, custom kernel or structuring element, and image archiving and retrieving.

Methods	Description
AssignMemory	Allocates the Image control using the memory at the specified location.
Bayer	Decode the color information from a single-band, Bayer encoded image.
Clear	Clears the image.
ChildRegion.Mode	Returns or sets the mode of operation of a child image.
Clone	Clones and returns an image identical to the specified image.
Copy	Copies data from a source image into the image.
CopyClip	Copies data from a source image, starting from the specified offset and clipping data that falls outside the image.
CopyConditional	Conditionally copies data from a source image to the image. Each image pixel is overwritten only if the corresponding pixel in the conditional image satisfies the specified condition. Other pixels are unchanged.
CopyMask	Copies data, with a bit-plane mask, from a source image to the image. Each image bit is changed only if it has a corresponding non-zero bit in the mask.
CopyRegion	Copies data from a region of a source image into the specified region of the image.
CopyToClipboard	Copies data from the current source image into the clipboard.

Image control (contined)

Methods	Description
FileInquire	Inquire about the data in a file.
Get	Copies a region of image data into a specified user-array. This region can be contained in one band or all bands of the image.
GetLine	Reads the pixels of a theoretical line between specified coordinates, counts them, and stores them in a specified array.
Load	Loads the image from a file, using the format specified by the FileFormat property, into the Image control.
PasteFromClipboard	Pastes image data (in CF_DIB format) from the clipboard into an Image control.
Put	Puts data from a specified array into a region of the image.
PutLine	Overwrites a specified series of pixels within specified coordinates, along a theoretical line with the data in a user-array.
Save	Saves the image data into a file, using the format specified by the FileFormat property.
ShowPropertyPages	Opens the specified property pages of the Image control in a window.

Properties	Description
Calibration	Returns or sets the Calibration control to associate to the image.
ChildRegion	Allows you to specify or determine the child image's region within its parent.
CompressionType	Returns or sets the type of compression, if any, to apply to the image data.
FileFormat	Returns or sets the file format used when saving or loading the image.
Format	Returns or sets the image's internal data format.
JPEGAlgorithm	Allows you to specify or determine the parameters of the JPEG algorithm used for the compression.
JPEG2000Algorithm	Allows you to specify or determine the parameters of the JPEG2000 algorithm used for the compression.
LUT	Allows you to specify or determine the custom LUT associated with the image.
NumberOfBands	Returns or sets the number of color bands of the Image control.
SizeX, SizeY	Returns or sets the image's X or Y width.

Events	Description
ContentModified	Occurs when the content of the image is modified.

ImageProcessing control

Used to perform filtering, morphological, point-to-point, segmentation, and statistical operations on an image. This control also includes geometric, color space, and domain transforms, as well as other image processing primitives.

Methods	Description
AbsoluteValue	Performs a point-to-point absolute value operation on an image.
Add	Performs a point-to-point addition operation using two source images.
And	Performs a point-to-point bitwise AND operation using two source images.
Binarize	Performs a point-to-point binary thresholding operation on an image.
CalculateStats	Calculates a variety of statistics on an image.
Clip	Performs a point-to-point clipping operation on an image.
Close	Performs a morphological closing operation on an image.
ConnectMap	Performs a 3x3 binary connectivity mapping operation on an image.
Convolve	Performs a custom convolution operation on an image.
Convert	Performs a color conversion operation on an image.
CountDifferences	Counts the number of pixels that differ between two images.
Dilate	Performs a morphological dilation operation on an image.
DiscreteCosineTransform	Performs a discrete cosine transform operation on an image.
Distance	Performs a distance transformation operation on an image.
Divide	Performs a point-to-point division operation using two source images.
EdgeDetect	Performs an edge detection operation and produces a gradient intensity and/or gradient angle image.
EdgeDetect1	Applies an edge detection filter, which is a fast approximation of a Sobel filter, to an image.
EdgeDetect2	Applies an edge detection filter, which is a fast approximation of a Prewitt filter, to an image.
Erode	Performs a morphological erosion operation on an image.
FastFourierTransform	Performs a fast Fourier transform operation on an image.
FindExtremes	Finds an image's extremes (minimum and/or maximum pixel values).
Flip	Performs a horizontal or vertical image-flipping operation.
Histogram	Generates the intensity histogram of an image.
HistogramEqualize	Performs a histogram equalization operation on an image or generates the LUT required to perform this operation.
HitOrMiss	Performs a morphological hit or miss transformation on an image.
HorizontalEdge	Applies a horizontal edge detection filter on an image.
Label	Labels blobs in an image.
LaplacianEdge1	Applies a Laplacian edge 1 filter on an image.
LaplacianEdge2	Applies a Laplacian edge 2 filter on an image.

ImageProcessing control (continued)

Methods	Description
LocateEvents	Locates pixels corresponding to a specified criteria in an image.
LUTMap	Performs a point-to-point LUT mapping operation on an image
Match	Performs a morphological matching operation on an image.
Maximum	Performs a point-to-point maximum operation using two source images.
Minimum	Performs a point-to-point minimum operation using two source images.
Multiply	Performs a point-to-point multiply operation using two source images.
MultiplyAndAccumulate1	Performs a point-to-point multiply and accumulate 1 operation using the stated source images.
MultiplyAndAccumulate2	Performs a point-to-point multiply and accumulate 2 operation using the stated source images.
Nand	Performs a point-to-point NAND operation using two source images.
Negate	Performs a point-to-point negate operation on an image.
Nor	Performs a point-to-point NOR operation using two source images.
Not	Performs a point-to-point NOT operation on an image.
OffsetGain	Performs a per-pixel gain and offset correction on an image.
Open	Performs a morphological opening operation on an image.
Or	Performs a point-to-point OR operation using two source images.
PolarToRectangular	Performs a polar-to-rectangular transform.
Project	Projects a 2-D image into 1-D.
Rank	Performs a rank filter on an image.
RectangularToPolar	Performs a rectangular-to-polar transform.
Resize	Resizes an image in X and/or Y.
Rotate	Rotates an image around the specified center of rotation.
Sharpen1	Applies a sharpening filter, which places equal emphasis on all neighboring pixels.
Sharpen2	Applies a sharpening filter, which places emphasis only on horizontally and vertically touching neighbors.
Shen	Applies a Shen-Castan Infinite Support Exponential (IIR) filter on an image.
Shift	Performs a point-to-point bit shift operation on an image.
ShowPropertyPages	Opens the specified property pages of the Image Processing control in a window.
Smooth	Applies a smoothing filter on an image.
Subtract	Performs a point-to-point subtraction operation using two source images.
Thick	Thickens blobs in an image.
Thin	Thin blobs in an image.
Translate	Translates an image in X and/or Y with sub-pixel accuracy.

ImageProcessing control (continued)

Methods	Description
VerticalEdge	Applies a vertical edge detection filter, which computes the absolute value of the vertical derivative of the image.
Warp	Warps an image.
WarpParameters.GenerateLUT	Generates the LUT entries for a 3x3 matrix-defined warping.
WarpParameters.MapQuadrilateralToRectangle	Generates coefficients for a perspective warping that maps a quadrilateral onto a rectangle.
WarpParameters.MapRectangleToQuadrilateral	Generates coefficients for a perspective warping that maps a rectangle onto a quadrilateral.
WarpParameters.ResetCoefficients	Resets the matrix of warping coefficients.
WarpParameters.RotateCoefficients	Generates warp coefficients for a counter-clockwise rotation.
WarpParameters.ScaleCoefficients	Generates warp coefficients for a scaling.
WarpParameters.ShearCoefficients	Generates warp coefficients for a shearing.
WarpParameters.TranslateCoefficients	Generates warp coefficients for a translation.
Watershed	Performs a watershed transform on an image.
WeightedAverage	Performs a point-to-point weighted average operation on an image.
Xnor	Performs a point-to-point XNOR operation on an image.
Xor	Performs a point-to-point XOR operation on an image.
ZoneOfInfluence	Performs a zone of influence operation on an image.

Properties	Description
Kernels	Returns the collection of kernels available to the ImageProcessing control, allowing access to its elements.
LUTs	Returns the collection of LUTs available to the ImageProcessing control, allowing access to its elements.
PolarParameters	Allows you to specify or determine the polar parameters of the image.
Results	Returns the collection of statistical image processing results obtained by the ImageProcessing control, allowing access to the collection's elements.
StructuringElements	Returns the collection of structuring elements available to the ImageProcessing control, allowing access to its elements.
WarpParameters	Allows you to specify or determine the coefficients or LUTs used to warp an image.

Events	Description
ResultsModified	Occurs after results have been modified.

Measurement control

Used to locate and measure edges or stripes within an image. Also used to take measurements between points, edges, or stripes. This control includes functions to save or restore markers (i.e., points, edges, or stripes).

Methods	Description
Calculate	Performs measurement calculations between two specified markers.
FindMarker	Finds an edge or stripe marker in the image.
Markers.Add	Adds a new marker to the Measurement Markers collection.
Markers.Item.Draw	Draws marker features in the destination image.
Markers.Item.SaveStream	Saves the characteristics of a measurement marker to a specified file or memory.
Markers.Load	Loads a marker from a file into the Measurement Markers collection.
Markers.LoadStream	Loads the characteristics of a previously saved measurement marker from a file or memory.
Markers.Remove	Removes a marker from the Measurement Markers collection.
Results.CalculateMaximum	Calculates the maximum value of a characteristic for all the found edges or stripes of a multiple marker.
Results.CalculateMean	Calculates the mean value of a characteristic for all the found edges or stripes of a multiple marker.
Results.CalculateMinimum	Calculates the minimum value of a characteristic for all the found edges or stripes of a multiple marker.
Results.CalculateStandardDeviation	Calculates the standard deviation of a characteristic for all the found edges or stripes of a multiple marker.
Results.Item.Draw	Draws specific result features in the destination image.
Results.Item.Edge1.Draw	Draws specific result features of the first edge of the found stripe, in the destination image.
Results.Item.Edge2.Draw	Draws specific result features of the second edge of the found stripe, in the destination image.
ShowPropertyPages	Opens the specified property pages of the Measurement control in a window.

Properties	Description
Markers	Returns the collection of markers of the Measurement control, allowing access to the collection's elements.
Markers.Item.Contrast	Allows you to specify the marker's expected contrast.
Markers.Item.EdgeStrength	Allows you to specify the marker's expected edge strength.
Markers.Item.EdgeThreshold	Returns or sets the edge value beneath which a grayscale variation is not considered an edge.
Markers.Item.FilterSmoothness	Returns or sets the degree of smoothness (strength of denoising) applied to the internal projection buffer of the search region during the edge extraction.
Markers.Item.NumberOfInsideEdges	Allows you to specify the expected number of inside edges of a stripe marker.
Markers.Item.Polarity	Allows you to specify the marker's expected polarity.
Markers.Item.Position	Allows you to specify the expected position of the marker.

Measurement control (continued)

Properties	Description
Markers.Item.SearchRegion	Allows you to specify the region within the target image to search for the marker.
Markers.Item.SearchRegion.Angle	Allows you to specify the angle, or angular range, of the search region.
Markers.Item.Spacing	Allows you to specify the expected inter-edge or inter-stripe spacing of a multiple marker.
Markers.Item.Width	Allows you to specify the expected width of a stripe marker.MeasurementList.
Markers.Item.Spacing	Allows you to specify the expected inter-edge or inter-stripe spacing of a multiple marker.
PixelAspectRatio	Allows you to specify or determine the target image's pixel aspect ratio.
Results	Returns the collection of measurement results obtained with the Measurement control, allowing access to the collection's elements.
Results.EdgeValues	Returns the collection of edge values available to the Measurement control, allowing access to its elements.
Results.Item.Edge1	Returns the measurement results for the first edge of the occurrence of the stripe marker.
Results.Item.Edge2	Returns the measurement results for the second edge of the occurrence of the stripe marker.

Events	Description
ResultsModified	Occurs after results have been modified.

ModelFinder control

Use geometric features (i.e., contours) to find models in an image. This control includes functions to define models, control search strategy, and save and restore a model.

Methods	Description
Find	Searches for the models of the ModelFinder control in the target image.
Load	Loads a previously saved ModelFinder control from a file.
LoadStream	Loads the settings of a previously saved ModelFinder control from a file or memory.
FindInEdgeFinderResults	Searches for the models of the ModelFinder control, in the results of an EdgeFinder control.
Models.AddCircleModel	Adds a model of a circle to the collection of models.
Models.AddCrossModel	Adds a model of a cross to the collection of models.
Models.AddAddDiamondModel	Adds a model of a diamond to the collection of Model Finder models.
Models.AddDxfModel	Defines a model from a CAD DXF file and adds it to the collection of models.
Models.AddEdgeModel	Adds a model defined from the edge extraction results obtained by a specified EdgeFinder control.
Models.AddEllipseModel	Adds a model of an ellipse to the collection of models.
Models.AddFromMerge	Adds a model from the active edges of two other models in the control, to the collection of Model Finder models.
Models.AddFromResult	Adds a model from the edges of one or all the results, to the collection of Model Finder models.
Models.AddImageModel	Adds a new image type model, from the specified image, to the collection of ModelFinder models.
Models.AddRectangleModel	Adds a model of a rectangle to the collection of models.
Models.AddRingModel	Adds a model of a ring to the collection of models.
Models.AddSquareModel	Adds a model of a square to the collection of models.
Models.AddTriangleModel	Adds a model of a triangle to the collection of Model Finder models.
Models.Item.Draw	Draws specified model features in the destination image.
Models.Item.Mask	Masks regions of the specified model.
Models.Remove	Removes a model from the collection of model finder models.
Preprocess	Preprocesses the ModelFinder control. This method extracts the active edges of models contained within the ModelFinder control and sets internal search settings so that future search will be optimized for speed and robustness.
Results.Item.Draw	Draws the specified features of the occurrence in the destination image at the found position, angle, and scale.
Save	Saves the settings of the ModelFinder control to disk.
SaveStream	Saves the settings of a ModelFinder control to a specified file or memory.
ShowPropertyPages	Opens the specified property pages of the ModelFinder control in a window.

ModelFinder control (Continued)

Properties	Description
FilterMode	Returns or sets the filtering mode to use for the edge extraction.
ModelFinderType	Returns or sets the type of search algorithm used by the ModelFinder control.
Models	Returns the collection of models available to the ModelFinder control, allowing access to its elements.
Models.Accuracy	Returns or sets the accuracy required when searching for ModelFinder models.
Models.Count	Returns the number of elements in the collection of ModelFinder models.
Models.DetailLevel	Returns or sets the level of details to extract from the model source and target images.
Models.DetailLevel	Returns or sets the level of details to extract from the model source and target images.
Models.Item.Chains	Returns the collection of chains associated with the active edges of the model, allowing access to the collection's elements.
Models.Item.Position	Allows you to specify or determine the position range in the target where positions for model occurrences can be found.
Models.SearchPositionEnabled	Returns or sets whether to perform calculations specific to position-range search strategies.
Models.SearchScaleEnabled	Returns or sets whether a search through a range of scales is enabled.
Models.SharedEdges	Returns or sets whether sharing of edges between occurrences is enabled.
Models.SmoothnessLevel	Returns or sets the degree of smoothing applied to the model source and target images.
Models.Speed	Returns or sets the required search speed.
Models.TotalNumberOfOccurrences	Returns or sets the maximum number of all model occurrences (for all models within the ModelFinder control together) to find in the target image.
Results	Returns the collection of ModelFinder results obtained by the ModelFinder control after a call to the Find method, allowing access its elements.
Results.Item.ModelChains	Returns the collection of model chains calculated for the model, allowing access to the results of each chain.
Results.TargetChains	Returns the collection of chains in the target image, allowing access to the results of each chain.

PatternMatching control

Used to locate patterns in an image using normalized grayscale correlation (NGC). This control includes functions to define a pattern, control search strategy, and save and restore a pattern.

Methods	Description
FindModel	Finds the specified pattern matching model(s) in the target image.
Models.AddAutomatic	Automatically adds a new unique model of the specified type to the collection.
Models.Item.Draw	Draw specific features of the model in the destination image.
Models.ImportDontCareImage	Sets the model's "don't care" pixels.
Models.Item.Preprocess	Preprocesses the pattern matching model. This trains the PatternMatching control to search for a model in the most efficient manner.
Models.Save	Saves the model to disk.
Models.Load	Loads the model from disk.
Results.Item.Draw	Draw specific features of the result occurrence in the destination image.
ShowPropertyPages	Opens the specified property pages of the Pattern Matching control in a window.

Properties	Description
Models	Returns the collection of pattern matching models available to the PatternMatching control, allowing access to its elements.
Models.Item.AcceptanceThreshold	Returns or sets the minimum acceptance level for a match made with the specified model when it is sought in an image.
Models.Item.CertaintyThreshold	Returns or sets the match score at which an occurrence of the model is assumed, without looking for better matches elsewhere in the image.
Models.Item.NumberOfOccurrences	Returns or sets the number of model occurrences for which to search in the target image.
Models.Item.PositionAccuracy	Returns or sets the positional accuracy required when searching for the model.
Models.Item.SearchAlgorithm	Allows you to specify or determine the model's search algorithm properties.
Models.Item.SearchAngle	Allows you to specify or determine the model's angular search properties.
Models.Item.SearchRegion	Allows you to specify or determine the region in which the search will be performed.
Models.Item.Speed	Returns or sets the required search speed for the search.
MultipleModelMode	Returns or sets whether you can search for more than one model with the FindModel method.
Results	Returns the collection of pattern matching results obtained by the PatternMatching control, allowing access to its elements.

Events	Description
ResultsModified	Occurs after results have been modified.

StringReader control*

Feature-based character recognition. This control supports multiple user-defined grammar rules and multi-font definition in a single context.

Methods	Description
Fonts.Add	Adds a font to the StringReader control's font collection.
Fonts.Remove	Removes the Font at the specified index from StringReader's fonts collection.
Load	Loads the StringReader control information from a file and allocates it on the system specified by the OwnerSystem Property.
LoadStream	Loads the settings of a previously saved StringReader control from a file or memory.
Preprocess	Prepares the string reader control, it's fonts and string models for reading.
Read	Perform a read operation in the specified target image.
Save	Saves the StringReader Control to a file.
SaveStream	Saves the settings of a StringReader control to a specified file or memory.
Models.Load	Loads the model from disk.
Results.Item.Draw	Draw specific features of the result occurrence in the destination image.
ShowPropertyPages	Opens the specified property pages of the Pattern Matching control in a window.

Properties	Description
CharacterEncodingType	Returns or sets the type of character encoding used by the string reader control.
Fonts	Returns the collection of fonts available to the StringReader control, allowing access to its elements.
Image	Returns or sets the image used as the target image for the string reader control.
LastDrawSizeX, Y	Returns the last Size X or Y needed by the last call to the Draw method.
MinimumContrast	Returns or sets the minimum contrast between a character of the target image and it's background.
Results	Returns the collection of StringReader results.
Results.Characters.Item. AspectRatio	Returns the aspect ratio of the character.
Results.Characters.Item. ConsecutiveSpaces	Returns the number of consecutive spaces that can be inserted between this character and the following character in the string.
Results.Characters.Item.PositionX, Y	Returns the position X or Y of the character.
Results.Characters.Item.Scale	Returns the scale of the character.
Results.Characters.Item. TransformationCoefficients	Returns the forward or reverse transformation coefficients 'a', 'b', 'c', 'd', 'e', or 'f'.
Results.Characters.Item.Value	Returns the character read.

* Available as of Processing Pack 1.

StringReader control (continued)*

Properties	Description
Results.Characters.Item.AspectRatio	Returns the aspect ratio of the character.
Results.Characters.Item.AspectRatio	Returns the aspect ratio of the character.
Results.Characters.Item.AspectRatio	Returns the aspect ratio of the character.
Results.DrawingParameters.RelativeOriginX, Y	Returns or sets the relative x or y offset attached to the origin of the destination image when drawing results.
Results.DrawingParameters.ScaleX, Y	Returns or sets the scale in the x direction attached to the destination image when drawing results.
Results.Strings.Item. FormattedValue	Returns the formatted string.
SeparatorCharacter	Returns or sets the character to be used as a string separator within the formatted text read.
ScoreNumber	Returns or sets the number of the score to evaluate on the sorted candidates.
ScoreType	Returns or sets the type of user score to use.
SpaceCharacter	Returns or sets the character to be used as a space character within the formatted text read.
Speed	Returns or sets the StringReader's search and read speed.
StreamSize	Returns the number of bytes required to stream the StringReader control.
StringModels	Returns the collection of string models available to the StringReader control, allowing access to its elements.
StringReaderType	Returns or sets the StringReader control's type.

Events	Description
ResultsModified	Occurs after results have been modified.

* Available as of Processing Pack 1.

Threading control

Used for the allocation of ActiveMIL thread contexts and synchronization events, including control over the created ActiveMIL thread contexts and events, inquire about various settings, and synchronize execution of multiple threads.

Methods	Description
Events.AddFromExternalEvent	Adds a new ActiveMIL event to the collection of threading events by mapping it to an existing ActiveMIL event.
Events.Item.SetState	Sets the state of the ActiveMIL threading event.
Wait	Performs a wait operation on an ActiveMIL selectable thread or ActiveMIL threading event.
ShowPropertyPages	Opens the specified property pages of the Threading control in a window.
Threads.Item.CommandsAbort	Aborts all the ActiveMIL commands queued in the selectable thread.

Properties	Description
Events.Item.AutoReset	Returns whether the threading event is reset automatically.
Threads.Item.Priority	Returns or sets the priority status of the ActiveMIL selectable thread.
Threads.Item.SynchronizationMode	Returns or sets the synchronization mode of the ActiveMIL selectable thread.
Threads	Returns the collection of ActiveMIL selectable threads available to the Threading control, allowing access to its elements.

Corporate headquarters:

Canada and U.S.A.
Matrox Electronic Systems Ltd.
1055 St. Regis Blvd.
Dorval, Quebec H9P 2T4
Canada
Tel: +1 (514) 685-2630
Fax: +1 (514) 822-6273

For more information, please call: 1-800-804-6243 (toll free in North America) or (514) 822-6020
or e-mail: imaging.info@matrox.com or http://www.matrox.com/imaging

matrox