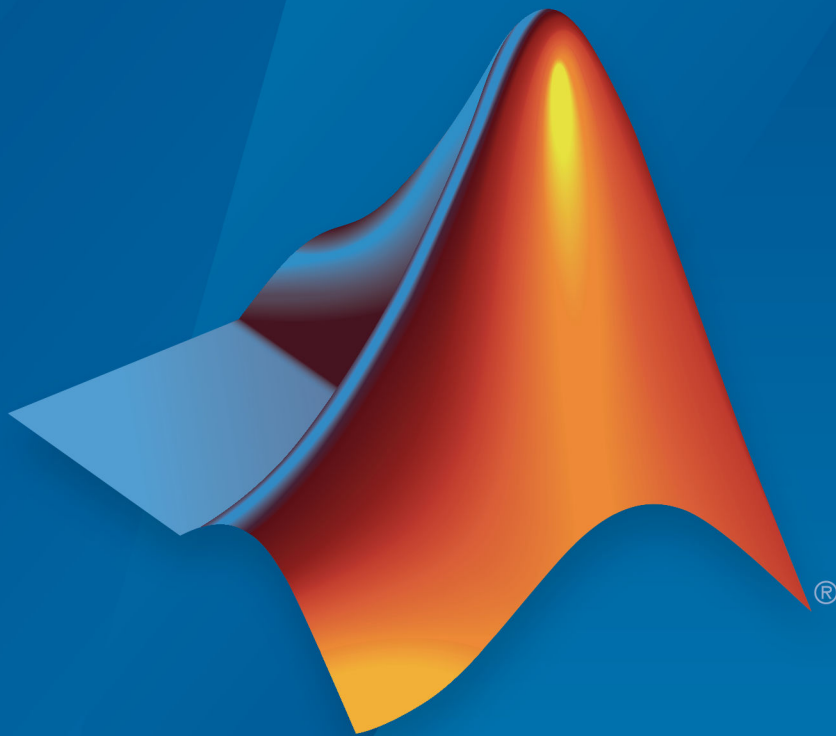Computer Vision System Toolbox™ Release Notes

MATLAB®

MathWorks®

How to Contact MathWorks

Latest news: www.mathworks.com

Sales and services: www.mathworks.com/sales_and_services

User community: www.mathworks.com/matlabcentral

Technical support: www.mathworks.com/support/contact_us

Phone: 508-647-7000

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098

Contents

R2016b

R2016a

R2015b

R2015a

R2014b

R2014a

R2012b

<div style="text-align:right">

R2012a

</div>

R2011b

R2011a

R2010b

R2010a

R2017b

Version: 8.0

New Features

Compatibility Considerations

Semantic Segmentation Using Deep Learning: Classify pixel regions in images, evaluate, and visualize segmentation results

Several new features to support semantic segmentation using deep learning techniques:

- `semanticseg`: Perform semantic image segmentation on images and image collections.
- `segnetLayers` and `fcnLayers`: Create SegNet and a fully convolutional network (FCN) segmentation networks.
- `pixelLabelImageSource`: Provides training data for semantic segmentation networks. Additionally supports several on-the-fly data augmentation techniques during training.
- `pixelLabelDatastore`: Provides a data store object that can be used to read pixel label data.
- `evaluateSemanticSegmentation`: Evaluate semantic segmentation results using intersection-over-union (IoU) and other commone metrics.
- `crop2dLayer`: Provides a layer for center cropping an input feature map.
- `pixelClassificationLayer`: Creates a pixel classification layer.

Image Labeling App: Interactively label individual pixels for semantic segmentation and label regions using bounding boxes for object detection

Use the **Image Labeler** app for interactive image labeling. You can label rectangular regions of interest (ROI) for object detection, pixels for semantic segmentation, and scenes for image classification. You can also define and execute custom label automation algorithms with the app.

Fisheye Camera Calibration: Calibrate fisheye cameras to estimate intrinsic camera parameters

The following functions support calibration of a wide-angle fisheye camera:

- `fisheyeCalibrationErrors`
- `fisheyeParameters`

- `fisheyeIntrinsics`
- `fisheyeIntrinsicsEstimationErrors`
- `undistortFisheyeImage` and `undistortFisheyePoints`

In addition, the `showExtrinsics` and `showReprojectionErrors` now accommodate fisheye data.

KAZE Features: Detect and extract KAZE features for object recognition or image registration workflows

Detect KAZE points from an image with the `detectKAZEFeatures` function, which returns a `KAZEPoints` object. Extract the KAZE features with the new KAZE method selection of the `extractFeatures` function.

Code generation for camera intrinsics

The `cameraIntrinsics` object now supports code generation.

Image Labeler app replaces Training Image Labeler app

Use the **Image Labeler** app in place of the Training Image Labeler app. The `trainingImageLabeler` function now opens the Image Labeler app.

Ground Truth Labeling Utitlities

The `groundTruth`, `groundTruthDataSource`, and `objectDetectorTrainingData` functions are added to support the image labeling.

Ground Truth Labeling Utilities	Description
`groundTruth`	Object for storing ground truth labels
`groundTruthDataSource`	Create a ground truth data source
`objectDetectorTrainingData`	Create training data from ground truth data for an object detector

Computer Vision Examples

·

· "Sensor Fusion Using Synthetic Radar and Vision Data" (Automated Driving System Toolbox)

· "Sensor Fusion Using Synthetic Radar and Vision Data" (Automated Driving System Toolbox)

R2017a

Version: 7.3

New Features

Bug Fixes

Compatibility Considerations

Deep Learning for Object Detection: Detect objects using Fast R-CNN and Faster R-CNN object detectors

Use `trainFastRCNNObjectDetector` to train a Fast R-CNN deep learning object detector. Use `trainFasterRCNNObjectDetector` to train a Faster R-CNN deep learning object detector. You can train a Faster R-CNN detector to detect multiple object classes. Also new this release are the `fastRCNNObjectDetector` and `fasterRCNNObjectDetector` RCNN support functions.

Object Detection Using ACF: Train object detectors using aggregate channel features

Use `trainACFObjectDetector` to train a classifier to recognize rigid objects. Use `acfObjectDetector` to detect objects in images.

Object Detector Evaluation: Evaluate object detector performance, including precision and miss-rate metrics

Use the `evaluateDetectionPrecision` to return the average precision to measure detection performance. Use the `evaluateDetectionMissRate` to evaluate the miss rate metric for object detection.

OpenCV Interface: Integrate OpenCV version 3.1.0 projects with MATLAB

Integrate OpenCV projects with MATLAB® using OpenCV version 3.1.0.

Object for storing intrinsic camera parameters

Use the `cameraIntrinsics` object to store information about a camera's intrinsic calibration parameters, including the lens distortion parameters.

Disparity function updated to fix inconsistent results between multiple invocations

In prior releases, the disparity function sporadically returned different results for depth estimation using `SemiGlobal` method.

Compatibility Considerations

When you use the `disparity` function's `SemiGlobal` method, the results will be different. Examine the results from your code carefully to see if you need to make any adjustments.

Improved algorithm to calculate intrinsics in Camera Calibration apps

This release improves the stability of estimating the principle point in Camera Calibration apps.

Compatibility Considerations

To reproduce prior results, you must use `estimateCameraParameters` function for camera calibration and do not specify the `ImageSize` property.

R2016b

Version: 7.2

New Features

Bug Fixes

Compatibility Considerations

Deep Learning for Object Detection: Detect objects using region-based convolution neural networks (R-CNN)

Use the `trainRCNNObjectDetector` function and the `rcnnObjectDetector` object to train an R-CNN deep learning object detector.

Structure from Motion: Estimate the essential matrix and compute camera pose from 3-D to 2-D point correspondences

Use the `estimateEssentialMatrix`, `estimateWorldCameraPose`, `extrinsicsToCameraPose`, `cameraPoseToExtrinsics` functions to estimate a 3-D structure of a scene from a set of 2-D images. It also adds a `worldToImage` method to the `cameraParameters` class to project world points into an image.

Point Cloud File I/O: Read and write PCD files using Point Cloud File I/O Functions

The `pcread` and `pcwrite` functions now support PCD (point cloud data) format files.

Code Generation for ARM Example: Detect and track faces on a Raspberry Pi 2 target

This release adds two examples that detail the steps for generating code for detecting and tracking faces on the Raspberry Pi 2 hardware.

- Detect Face (Raspberry Pi2)
- Track Face (Raspberry Pi2)

Visual Odometry Example: Estimate camera locations and trajectory from an ordered sequence of images

This release adds a visual odometry example, Monocular Visual Odometry, that details the steps for estimating camera locations and camera trajectory.

cameraPose function renamed to relativeCameraPose

The `cameraPose` function has been renamed to the more descriptive `relativeCameraPose`. Additionally, the function can now accept an essential matrix from the new `estimateEssentialMatrix` function.

New capabilities for Training Image Labeler app

You can now use the Training Image Labeler app to:

- Create a full-image region of interest (ROI).
- Add multiple ROI labels (categories).
- Import ROIs from a `MAT` file or from the workspace.
- Output a table if there are multiple ROI labels.

Train cascade object detector function takes tables and uses imageDatastore

The `trainCascadeObjectDetector` function can now take positive instances as a table or as a `struct` array. It can also take negative images using `imageDatastore`.

Project 3-D world points into image

The `cameraParameters` object now provides a `worldToImage` method that projects 3-D world points into an image.

Code generation support

The following functions and methods:

- `cameraPoseToExtrinsics`
- `extrinsicsToCameraPose`
- `worldToImage` method of the `cameraParameters` object
- `estimateEssentialMatrix`
- `estimateWorldCameraPose`

- `relativeCameraPose`

Plot camera function accepts a table of camera poses

The `plotCamera` function can now accept and plot a table of camera poses.

Eliminate 3-D points input from extrinsics function

The `extrinsics` function no longer accepts 3-D *x,y,z* points as an input. Instead, use the `estimateWorldCameraPose` function.

Compatibility Considerations

When you try to input *x,y,z* points, the `extrinsics` issues a warning.

Simpler way to call System objects

Instead of using the `step` method to perform the operation defined by a System object™, you can call the object with arguments, as if it were a function. The `step` method will continue to work. This feature improves the readability of scripts and functions that use many different System objects.

For example, if you create a `vision.Pyramid` System object named `gaussPyramid`, then you call the System object as a function with that name.

```
gaussPyramid = vision.Pyramid('PyramidLevel',2);
gaussPyramid(x);
```

The equivalent operation using the `step` method is:

```
gaussPyramid = vision.Pyramid('PyramidLevel',2);
step(gaussPyramid,x);
```

When the `step` method has the System object as its only argument, the function equivalent has no arguments. This function must be called with empty parentheses. For example, `step(sysobj)` and `sysobj()` perform equivalent operations.

R2016a

Version: 7.1

New Features

Bug Fixes

Compatibility Considerations

OCR Trainer App: Train an optical character recognition (OCR) model to recognize a specific set of characters

This release adds the OCR Trainer app.

Structure from Motion: Estimate the camera poses and 3-D structure of a scene from multiple images

This release adds a collection of functions and objects to support structure from motion.

- `bundleAdjustment`
- `pointTrack`
- `viewSet` with several supporting methods for finding tracks and storing camera poses.
- `triangulateMultiview`

Pedestrian Detection: Locate pedestrians in images and video using aggregate channel features (ACF)

This release adds the `detectPeopleACF` function to detect people in a scene.

Bundle Adjustment: Refine estimated locations of 3-D points and camera poses for the structure from motion (SFM) framework

This release adds the `bundleAdjustment` function to estimate camera poses and 3-D points simultaneously.

Multiview Triangulation: Triangulate 3-D locations of points matched across multiple images

This release adds the `triangulateMultiview` function to recover the location of a 3-D world point from its projections into 2-D images.

Rotate matrix to vector and vector to matrix

This release adds the `rotationMatrixToVector` and the `rotationVectorToMatrix` functions. These function implement the Rodrigues transform.

Select spatially uniform distribution of feature points

This release adds the `selectUniform` method to the `SURFPoints`, `cornerPoints`, and `BRISKPoints` objects.

Single camera and stereo camera calibration app enhancements

This release continues to enhance the single camera and stereo camera calibrator apps. The enhancements include:

- Ability to select multiple outlier images that correspond to a high mean reprojection error.
- Minimized analysis charts.
- Removed Viewing tab and placed viewing controls in the toolbar.
- Speeded up calibration for the Single Calibration App.

Point cloud from Kinect V2

This release adds support for extracting point clouds from Kinect® V2 using the `pcfromkinect` function.

Point cloud viewer enhancements

This release adds enhancements to the `pcshow`, `pcplayer`, and `pcshowpair` viewers. The functions now rotates the point cloud around the center of the axis and shows the rotation axis. The display of point clouds are downsampled for a large range of data. The downsampling is for display only, it does not modify the data. This change makes display time faster. The functions also now support `subplot`.

Support package for Xilinx Zynq-based hardware

The Computer Vision System Toolbox Support Package for Xilinx® Zynq®-Based Hardware supports verification and prototyping of vision algorithms on Zynq boards.

HDL Coder™ is required for customizing the algorithms running on the FPGA fabric of the Zynq device. Embedded Coder® is required for customizing the algorithms running on the ARM® processor of the Zynq device.

- Target your video processing algorithms to Zynq hardware from Simulink®
- Stream HDMI signals into Simulink to explore designs with real data
- Generate HDL vision IP cores using HDL Coder
- Deploy algorithms and visualize using HDMI output on a screen

For additional information, see Computer Vision System Toolbox Support Package for Xilinx Zynq-Based Hardware.

C code generation support

This release continues to add C code generation support to new and existing functions and objects.

- `rotationVectorToMatrix`
- `rotationMatrixToVector`
- `insertObjectAnnotation`

This release also adds new support for portable C code generation. You can generate C code for your specific target using any of the newly supported functions. The new support allows you to build the application for your target using a C++ compiler. The C++ compiler links to OpenCV (Version 2.4.9) libraries that you provide for the particular target. To build a standalone application, use `packNGo` with `'Hierarchical'` `packType`.

The newly supported functions for portable C code generation:

- `vision.CascadeObjectDetector`
- `detectBRISKFeatures`
- `detectFASTFeatures`
- `detectMSERFeatures`
- `disparity`
- `extractFeatures` for `BRISK`, `FREAK`, and `SURF` methods.
- `detectSURFFeatures`

- `vision.PeopleDetector`
- `vision.PointTracker`
- `matchFeatures`
- `opticalFlowFarneback`

Future removal warning of several System objects

The Computer Vision System Toolbox begins removal of overlapping functionality with equivalent functions.

Compatibility Considerations

Starting in this release, when you use any of the System objects listed in the table, MATLAB issues a warning. Replace the use of the System object with the corresponding function.

Computer Vision System Toolbox System object	Equivalent function
Analysis and Enhancement	
`vision.ContrasterAdjuster`	`imadjust`, `stretchlim`
`vision.CornerDetector`	`detectHarrisFeatures` , `detectMinEigenFeatures` ,`cornerPoints`
`vision.EdgeDetector`	`edge`
`vision.HistogramEqualizer`	`histeq`
`vision.OpticalFlow`	`opticalFlowLKDoG`,`opticalFlowLK`, `opticalFlowFarneback`, `opticalFlowHS`
`vision.BoundaryTracer`	`bwtraceboundary`,`bwboundaries`
Conversions	
`vision.Autothresholder`	`graythresh`, `multithresh`
`vision.ColorSpaceConverter`	`rgb2gray`, `rgb2ycbcr`, `makecform`, `applycform`
`vision.DemosaicInterpolator`	`demosaic`

Computer Vision System Toolbox System object	Equivalent function
vision.GammaCorrector	imadjust
vision.ImageComplementer	imcomplement
vision.ImageDataTypeConverter	im2double,im2single, im2uint8, im2int16, im2uint16
Filtering	
vision.ImageFilter	imfilter
vision.MedianFilter2D	medfilt2
Geometric Transformations	
vision.GeometricTransformer	imwarp
vision.GeometricTransformEstimator	fitgeotrans
vision.GeometricScaler	imresize
vision.GeometricRotator	imrotate
vision.GeometricTranslator	imtranslate
Sinks and Sources	
vision.BinaryFileWriter	No support
vision.BinaryFileReader	No support
Statistics	
vision.Histogram	imhist
vision.PSNR	psnr
Text & Graphics	
vision.MarkerInserter	insertMarker
vision.ShapeInserter	insertShape

Computer Vision System Toolbox System object	Equivalent function
vision.TextInserter	insertText
Transforms	
vision.HoughTransform	hough
Utilities	
vision.ImagePadder	padarray

R2015b

Version: 7.0

New Features

Bug Fixes

Compatibility Considerations

3-D Shape Fitting: Fit spheres, cylinders, and planes into 3-D point clouds using RANSAC

This release adds 3-D point cloud processing functions and classes to fit a sphere, cylinder, or plane to a point cloud.

- `pcfitsphere`
- `pcfitplane`
- `pcfitcylinder`
- `planeModel`
- `sphereModel`
- `cylinderModel`

Streaming Point Cloud Viewer: Visualize streaming 3-D point cloud data from sensors such as the Microsoft Kinect

This release adds the `pcplayer` function for visualizing streaming 3-D point cloud data.

Point Cloud Normal Estimation: Estimate normal vectors of a 3-D point cloud

This release adds the `pcnormals` function to estimate normal vectors for point clouds.

Farneback Optical Flow: Estimate optical flow vectors using the Farneback method

This release adds the `opticalFlowFarneback` object that allows you to compute optical flow. This object supports C code generation.

LBP Feature Extraction: Extract local binary pattern features from a grayscale image

This release adds the `extractLBPFeatures` function that extracts local binary patterns (LBP) from a grayscale image.

Multilanguage Text Insertion: Insert text into image data, with support for multiple languages

This release adds the ability to insert a TrueType font with the `insertText` function. Also new, is the `listTrueTypeFonts` function to list available TrueType fonts on your system.

3-D point cloud extraction from Microsoft Kinect

This release adds the `pcfromkinect` function to convert a Microsoft® Kinect depth and RGB image to a 3-*D* point cloud. To use this function you must have the Image Acquisition Toolbox™ installed.

3-D point cloud displays

This release adds the `pcshow` function for plotting a 3-D point cloud. The release also adds the `pcshowpair` function to visualize the differences between two point clouds.

Downsample point cloud using nonuniform box grid filter

This release adds a new syntax to the `pcdownsample` function which returns a downsampled point cloud using a nonuniform box grid filter. The box grid filter can be used as a preprocessing step for point cloud registration.

Compute relative rotation and translation between camera poses

This release adds the `cameraPose` function to compute the relative pose of a calibrated camera based on two views.

Warp block

This release adds the Warp block. Use the Warp block to apply projective or affine transforms to an image. You can use this block to transform the entire image or portions of the image with either a polygon or rectangular region of interest (ROI). This block replaces the Apply Geometric Transformation block.

Compatibility Considerations

Apply Geometric Transformation block will be removed in a future release. Use the Warp block instead.

GPU support for FAST feature detection

This release adds GPU acceleration for the `detectFASTFeatures` function. GPU acceleration for this function requires Parallel Computing Toolbox™.

Camera calibration optimization options

This release adds new properties to the `estimateCameraParameters` function that provide the ability to supply an initial guess for calibration parameters prior to optimization.

C code generation support

This release continues to add C code generation support to new and existing functions and objects.

- `cameraPose`
- `detectCheckerboardPoints`
- `extractLBPFeatures`
- `generateCheckerboardPoints`
- `insertText`
- `opticalFlowFarneback`

Examples for face detection, tracking, 3-D reconstruction, and point cloud registration and display

This release adds the following new featured examples:

- The Face Detection and Tracking Using Live Video Acquisition example shows how to automatically detect and track a face in a live video stream, using the KLT algorithm.
- The Tracking Pedestrians from a Moving Car example shows how to perform automatic detection and tracking of people in a video from a moving camera.

- The Structure From Motion From Two Views example shows you how to generate a point cloud from features matched between two images taken with a calibrated camera.
- The 3-D Point Cloud Registration and Stitching example shows how to combine multiple point clouds to reconstruct a 3-D scene using iterative closest point (ICP) algorithm.

Example using Vision HDL Toolbox for noise removal and image sharpening

This release, the Vision HDL Toolbox™ product adds the Noise Removal and Image Sharpening featured example. This example shows how to implement a front-end module of an image processing design. This front-end module removes noise and sharpens the image to provide a better initial condition for the subsequent processing.

Removed video package from Computer Vision System Toolbox

This release removes the use of the `video` package. It was replaced with the use of the `vision` package name.

Compatibility Considerations

Replace the use of the `video` name with `vision`.

Morphological System objects future removal warning

The Computer Vision System Toolbox begins removal of overlapping functionality with equivalent functions in the Image Processing Toolbox™.

Compatibility Considerations

Starting in this release, when you use any of the morphological System objects, MATLAB issues a warning. Replace the use of the System object with the corresponding Image Processing Toolbox function.

Computer Vision System Toolbox System object	Equivalent Image Processing Toolbox function
`vision.MorphologicalDilate`	`imdilate`
`vision.MorphologicalOpen`	`imopen`
`vision.MorphologicalClose`	`imclose`
`vision.MorphologicalErode`	`imerode`
`vision.MorphologicalBottomHat`	`imbothat`
`vision.MorphologicalTopHat`	`imtophat`
`vision.ConnectedComponentLabeler`	`bwlabel` or `bwlabeln` (n-D version)

No edge smoothing in outputs of undistortImage and rectifyStereoImages

The `undistortImage` and `rectifyStereoImages` functions were modified in version 7.0 of the Computer Vision System Toolbox (MATLAB R2015b).

Compatibility Considerations

Previous versions of these functions smoothed the edges of the output image. The current versions do not smooth the image borders in order to increase speed.

VideoFileReader play count

The `vision.VideoFileReader PlayCount` default value was changed from `inf` to `1`.

Compatibility Considerations

Any `vision.VideoFileReader` objects saved in previous versions and loaded into R2015b (or later) will loop continuously.

R2015a

Version: 6.2

New Features

Bug Fixes

Compatibility Considerations

3-D point cloud functions for registration, denoising, downsampling, geometric transformation, and PLY file reading and writing

This release adds 3-D point cloud processing functions. It also adds an object for storing a point cloud, and functions to read and write point cloud files in a PLY format.

- `pctransform`
- `pcregrigid`
- `pcmerge`
- `pcdownsample`
- `pcdenoise`
- `pointCloud`
- `pcread`
- `pcwrite`

Image search and retrieval using bag of visual words

This release adds functionality for searching large image collections. It adds the `retrieveImages`, `evaluateImageRetrieval`, and `indexImages` functions, and the `invertedImageIndex` object.

User-defined feature extractor for bag-of-visual-words framework

This release adds the ability to define a custom feature extractor to use with the `bagOfFeatures` framework.

C code generation for eight functions, including rectifyStereoImages and vision.DeployableVideoPlayer on Mac

This release adds C code generation support to eight functions, one System object, and one block. Additionally, the new optical flow functions support C code generation.

- `rectifyStereoImages`
- `reconstructScene`
- `undistortImage`

- `triangulate`
- `extrinsics`
- `cameraMatrix`
- `cameraParameters`
- `stereoParameters`
- `vision.DeployableVideoPlayer`
- To Video Display block on the Mac
- `opticalFlow`
- `opticalFlowHS`
- `opticalFlowLK`
- `opticalFlowLKDoG`

Mac support for vision.DeployableVideoPlayer and To Video Display block

This release enables the `vision.DeployableVideoPlayer` System object and the To Video Display block on the Mac. These players are capable of displaying high-definition video at high frame rates.

Compatibility Considerations

The player displays frames at the rate of the video input. Setting the `FrameRate` property has no effect on changing the rate of the display. The `FrameRate` property will be removed in a future release.

Plot camera figure in 3-D coordinate system

This release adds the `plotCamera` function. The function returns a camera visualization object to plot a camera figure in the current axes.

Line width for insertObjectAnnotation

This release introduces a `LineWidth` property for the `insertObjectAnnotation` function. Setting this property enables you to modify the line width of inserted annotations.

Upright option for extractFeatures

This release introduces the `Upright` property for the `extractFeatures` function. This property enables you to restrict extracted features to an upright orientation only.

Rotate integral images in integralImage, integralKernel, and integralFilter functions

This release introduces rotated summed area table (RSAT) support to the `integralImage`, `integralKernel`, and `integralFilter` functions.

Performance improvements

* `detectCheckerboardPoints` and `estimateCameraParameters` functions
* `integralFilter` function
* `vision.DeployableVideoPlayer` System object
* To Video Display block

Optical flow functions and object

This release adds the `opticalFlowHS`, `opticalFlowLK`, and `opticalFlowLKDoG` functions. These functions enable you to compute optical flow using the Horn-Schunck and Lucas-Kanade methods. This release also adds the `opticalFlow` object for storing optical flow velocity matrices. The functions and the object support C code generation.

Examples for image retrieval, 3-D point cloud registration and stitching, and code generation for depth estimation from stereo video

* Image Retrieval Using Customized Bag-of-Features

- 3-D Point Cloud Registration and Stitching
- Code Generation for Depth Estimation From Stereo Video

7

R2014b

Version: 6.1

New Features

Bug Fixes

Stereo camera calibration app

This release adds a stereo calibration app. You can use the app to calibrate a stereo camera.Use the `stereoCameraCalibrator` function to invoke the app. See the Stereo Calibration Using the Stereo Camera Calibrator App tutorial.

imageSet class for handling large collections of image files

This release adds the `imageSet` class for managing large collections of image files. Use the `imageSet partition` method to create subsets of image sets. You can use these subsets to provide training and validation images for classification.

Bag-of-visual-words suite of functions for image category classification

This release adds a suite of functions to support the bag-of-features workflow. The workflow allows you to manage your image collections and partition them into training and validation sets. It constructs a bag of visual words for use in image category classification. The training and classification includes support for Parallel Computing Toolbox.

- `imageSet`
- `bagOfFeatures`
- `trainImageCategoryClassifier` (requires Statistics and Machine Learning Toolbox™)
- `imageCategoryClassifier` (requires Statistics and Machine Learning Toolbox)

Approximate nearest neighbor search method for fast feature matching

This release provides updates to the `matchFeatures` function. The update replaces previous matching methods with `'Exhaustive'` and `'Approximate'` Nearest Neighbor methods. It also adds the `Unique` match logical property to only return unique matches from the input feature set.

As a result of this update, the following methods and properties were removed:

- `'NearestNeighborRatio'`, `'NearestNeighborSymmetric'`, and `'Threshold'` matching methods

- 'normxcorr' normalized cross-correlation metric and the 'Prenormalized' properties

Use the following new methods to match the behavior of the removed properties.

Previous Match Method	Set New Match Method
'NearestNeighborRatio'	Set the `Method` property to 'Exhaustive' and the `Unique` property to `false`.
'NearestNeighborSymmetric'	Set the `Method` property to 'Exhaustive', the `Unique` property to `true`, and the `MaxRatio` to 1.

3-D point cloud visualization function

This release adds the `showPointCloud` function for plotting point clouds.

3-D point cloud extraction from Kinect

This release adds the `depthToPointCloud` function to convert a Kinect depth image to a 3-*D* point cloud. This function requires the Image Acquisition Toolbox.

Kinect color image to depth image alignment

This release adds the `alignColorToDepth` function for registering a Kinect color image to a depth image. This function requires the Image Acquisition Toolbox.

Point locations from stereo images using triangulation

This release adds the `triangulate` function. You can use this function to find 3-D locations of matching points in stereo images.

Red-cyan anaglyph from stereo images

This release adds the `stereoAnaglyph` function. Use this function to combine stereo images to form an anaglyph, which can be viewed with red-blue stereo glasses.

Point coordinates correction for lens distortion

This release adds the `undistortPoints` function. Use this function to remove the effects of lens distortion from individual point locations.

Camera projection matrix

This release adds the `cameraMatrix` function. You can use the matrix returned by this function to project 3-D world points in homogeneous coordinates into an image.

Calculation of calibration standard errors

This release adds the ability to return the standard errors incurred during the camera calibration process. The `estimateCameraParameters` function returns the errors. You can use the errors to evaluate the quality of the camera calibration. You can return errors for both single and stereo camera calibration.

Live image capture in Camera Calibrator app

You can now do live camera calibration using the Camera Calibrator app. The new **Image Capture** tab allows you to bring live images from USB Webcams into the app. Previously, you had to save your images to disk and manually add them into the app.

The image capture functionality in the Camera Calibrator app allows you to:

- Capture live images from USB Webcams
- Browse the captured images
- Save acquired images
- Integrate between image acquisition and calibration
- Control camera properties, such as brightness and contrast.

Use the `cameraCalibrator` function to open the app. Then select **Add Images > From camera** to open the **Image Capture** tab. Select your device, set any properties, and define the capture settings. You can then capture images and calibrate the camera.

Region of interest (ROI) copy and paste support for Training Image Labeler app

This release adds the ability to copy-and-paste regions of interest within the Training Image Labeler app.

Non-maximal suppression of bounding boxes for object detection

This release adds the `bboxOverlapRatio` and the `selectStrongestBbox` functions. Use `bboxOverlapRatio` to compute the overlap ratio between pairs of bounding boxes. Use `selectStrongestBbox` to select the strongest bounding boxes from overlapping clusters.

Linux support for deployable video player

This release adds Linux® support for the To Video Display block and the `vision.DeployableVideoPlayer` System object. This added support includes the ability to generate code.

GPU support for Harris feature detection

This release adds GPU acceleration for the `detectHarrisFeatures` function. GPU acceleration for this function requires Parallel Computing Toolbox.

Extended language support package for optical character recognition (OCR)

This release adds the ability to download additional language support for the `ocr` function. You can use the `visionSupportPackages` function to download the language support package.

Support package for OpenCV Interface

This release adds a support package to help you integrate your OpenCV C++ code into MATLAB. It lets you build MEX files that calls OpenCV functions. You can use the `visionSupportPackages` function to download the OpenCV Interface support package.

Convert format of rectangle to a list of points

This release adds the `bbox2points` function. You can use this function to convert a rectangle, specified as [*x*, *y*, *width*, *height*], into a list of [*x*, *y*] points.

Bag-of-visual-words, stereo vision, image stitching, and tracking examples

This release adds several new examples.

- Pedestrian tracking from a moving car
- Image classification using bag-of-visual-words workflow
- Face tracking from a web cam
- Evaluate camera calibration results
- Image stitching
- Depth estimation from a stereo video
- Code generation with PackNGo

R2014a

Version: 6.0

New Features

Bug Fixes

Compatibility Considerations

Stereo vision functions for rectification, disparity calculation, scene reconstruction, and stereo camera calibration

This release adds a suite of stereo vision algorithms to the Computer Vision System Toolbox.

- `rectifyStereoImages` for stereo rectification.
- `reconstructScene` for computing dense 3-D reconstruction based on a disparity map.
- `extrinsics` for computing location of a calibrated camera.
- `stereoParameters` object for storing stereo system parameters.
- `detectCheckerboardPoints` extended to support stereo calibration
- `disparity` adds new method for semi-global block matching.
- `estimateCameraParameters` extended to calibrate stereo cameras.
- `cameraParameters` object for storing camera parameters.
- `showExtrinsics` extended to support stereo cameras.
- `showReprojectionErrors` extended to support stereo cameras.

Compatibility Considerations

This release modifies the `disparity` function's default method for block matching. The new `SemiGobal` default method may produce different results in code created that used the previous `BlockMatching` default method. To obtain the same results, set the `'Method'` property to `'BlockMatching'`.

Optical character recognition (OCR)

This release adds the `ocr` function and `ocrText` object. You can use the `ocr` function to recognize text using optical character recognition. The `ocrText` object stores optical character recognition results.

Binary Robust Invariant Scalable Keypoints (BRISK) feature detection and extraction

This release adds the `detectBRISKFeatures` function. You can use the Binary Robust Invariant Scalable Keypoints (BRISK) algorithm to detect multi-scale corner features.

This release also adds the `BRISKPoints` object to store the BRISK detection results. This release adds BRISK descriptor to the `extractFeatures`.

App for labeling images for training cascade object detectors

This release adds a Training Image Labeler app. The app can be used to select regions of interest in images for the purpose of training a classifier. You can invoke the app by using the `trainingImageLabeler` function. See the Label Images for Classification Model Training tutorial.

C code generation for Harris and minimum eigenvalue corner detectors using MATLAB Coder

This release adds C code generation support for the `detectHarrisFeatures` and `detectMinEigenFeatures` functions. This release also adds C code generation to the `estimateGeometricTransform` function.

Line width control for insertShape function and Draw Shapes block

This release adds line thickness control to the `insertShape` function and the Draw Shapes.

Replacing vision.CameraParameters with cameraParameters

This release replaces the `vision.CameraParameters` object with the `cameraParameters` object. The new object contains identical functionality.

Compatibility Considerations

You must replace the `vision.CameraParameters` with `cameraParameters` object in your code. If you attempt to create a `vision.CameraParameters` object, MATLAB returns an error.

Output view modes and fill value selection added to undistortImage function

This release adds new output view modes and fill value selection to the `undistortImage` function. You can control the output view size by setting the `OutputView` property. You can also set the fill value with the `FillValues` property.

Generated code optimized for the matchFeatures function and vision.ForegroundDetector System object

This release provides generated code optimization for the `matchFeatures` function and the `vision.ForegroundDetector` System object on a Windows®, Linux, Mac OS platform.

Merging mplay viewer into implay viewer

This release merges the `mplay` viewer function from the Computer Vision System Toolbox into the `implay` function in Image Processing Toolbox.

Compatibility Considerations

Use the `implay` function with functionality identical to `mplay`. The `mplay` function will be removed in a future release.

MPEG-4 and JPEG2000 file formats added to vision.VideoFileWriter System object and To Multimedia File block

This release adds support for writing MPEG-4 and JPEG 2000 file formats with the `vision.VideoFileWriter` object and the To Multimedia File block.

Region of interest (ROI) support added to detectMSERFeatures and detectSURFFeatures functions

This release adds region of interest (ROI) support to the `detectMSERFeatures` and `detectSURFFeatures` functions.

MATLAB code script generation added to Camera Calibrator app

This release adds MATLAB code script generation to the Camera Calibrator app.

Featured examples for text detection, OCR, 3-D reconstruction, 3-D dense reconstruction, code generation, and image search

This release the Computer Vision System Toolbox adds several new featured examples:

- Automatically Detect and Recognize Text in Natural Images
- Image Search using Point Features
- Recognize Text Using Optical Character Recognition (OCR)
- Code Generation for Feature Matching and Registration (updated)
- Stereo Calibration and Scene Reconstruction
- Sparse 3-D Reconstruction From Multiple Views

Play count default value updated for video file reader

This release the Computer Vision System Toolbox modifies the default value from `inf` to `1` for the `PlayCount` property of the VideoFileReader System object. This change allows proper functionality while using the `isDone` method.

R2013b

Version: 5.3

New Features

Bug Fixes

Compatibility Considerations

Camera intrinsic, extrinsic, and lens distortion parameter estimation using camera calibration app

This release adds a camera calibration app. The app can be used to estimate camera intrinsic and extrinsic parameters, and to compute parameters needed to remove the effects of lens distortion from an image. You can invoke the calibrator using the `cameraCalibrator` function. See the Find Camera Parameters with the Camera Calibrator tutorial.

Camera calibration functions for checkerboard pattern detection, camera parameter estimation, correct lens distortion, and visualization of results

This release adds a suite of functions that, when used together, provide a workflow to calibrate a camera:

- `detectCheckerboardPoints`
- `estimateCameraParameters`
- `generateCheckerboardPoints`
- `showExtrinsics`
- `showReprojectionErrors`
- `undistortImage`
- `vision.CameraParameters`

Histogram of Oriented Gradients (HOG) feature extractor

This release adds the `extractHOGFeatures` descriptor function. The extracted features encode local shape information from regions within an image. You can use this function for many tasks including classification, detection, and tracking.

C code generation support for 12 additional functions

This release adds C code generation support for several Computer Vision System Toolbox functions, classes, and System objects.

- `extractHOGFeatures`

- `extractFeatures`
- `detectSURFFeatures`
- `disparity`
- `detectMSERFeatures`
- `detectFASTFeatures`
- `vision.CascadeObjectDetector`
- `vision.PointTracker`
- `vision.PeopleDetector`
- `MSERRegions`
- `cornerPoints`
- `SURFPoints`

System objects matlab.system.System warnings

The System object base class, `matlab.system.System` has been replaced by `matlab.System`. If you use `matlab.system.System` when defining a new System object, a warning message results.

Compatibility Considerations

Change all instances of `matlab.system.System` in your System objects code to `matlab.System`.

Restrictions on modifying properties in System object Impl methods

When defining a new System object, certain restrictions affect your ability to modify a property.

You cannot use any of the following methods to modify the properties of an object:

- `cloneImpl`
- `getDiscreteStateImpl`
- `getDiscreteStateSpecificationImpl`
- `getNumInputsImpl`

- `getNumOutputsImpl`
- `getOutputDataTypeImpl`
- `getOutputSizeImpl`
- `isInputDirectFeedthroughImpl`
- `isOutputComplexImpl`
- `isOutputFixedSizeImpl`
- `validateInputsImpl`
- `validatePropertiesImpl`

This restriction is required by code generation, which assumes that these methods do not change any property values. These methods are validation and querying methods that are expected to be constant and should not impact the algorithm behavior.

Also, if either of the following conditions exist:

- You plan to generate code for the object
- The object will be used in the MATLAB System block

you cannot modify tunable properties for any of the following runtime methods:

- `outputImpl`
- `processTunedPropertiesImpl`
- `resetImpl`
- `setupImpl`
- `stepImpl`
- `updateImpl`

This restriction prevents tunable parameter updates within the object from interfering with updates from outside the generated code. Tunable parameters can only be changed from outside the generated code.

Compatibility Considerations

If any of your class definition files contain code that changes a property in one of the above `Impl` methods, move that property code into an allowable `Impl` method. Refer to the System object `Impl` method reference pages for more information.

R2013a

Version: 5.2

New Features

Bug Fixes

Compatibility Considerations

Cascade object detector training using Haar, Histogram of Oriented Gradients (HOG), and Local Binary Pattern (LBP) features

This release adds the `trainCascadeObjectDetector` function for Haar, Histogram of Oriented Gradients (HOG), and Local Binary Pattern (LBP) features. The function creates a custom classification model to use with the `vision.CascadeObjectDetector` cascade object detector.

Fast Retina Keypoint (FREAK) algorithm for feature extraction

This release adds the Fast Retina Keypoint (FREAK) descriptor algorithm to the `extractFeatures` function. This function now supports the FREAK method for descriptor extraction.

Hamming distance method for matching features

This release adds the Hamming distance method to the `matchFeatures` function in support of binary features produced by descriptors such as the FREAK method for extraction. It also adds the new `binaryFeatures` object, which is an output of the `extractFeatures` function and serves as an input to the `matchFeatures` function.

Multicore support in matchFeatures function and ForegroundDetector System object

This release brings multicore performance improvements for the `matchFeatures` function and the `vision.ForegroundDetector` detector.

Functions for corner detection, geometric transformation estimation, and text and graphics overlay, augmenting similar System objects

This release adds several new functions. For corner detection, the new `detectHarrisFeatures`, `detectMinEigenFeatures`, and `detectFASTFeatures` functions. The `insertText`, `insertMarker`, and `insertShape` functions for inserting text, markers, and shapes into images and video. Lastly, the `estimateGeometricTransform` function for estimating a geometric transform from putatively matched point pairs.

Error-out condition for old coordinate system

This release ends support for the row-column coordinate system for the Computer Vision System Toolbox algorithms. All blocks are replaced with blocks using [x y] coordinates, and all functions and System objects are updated to use the one-based [x y] convention. Using any MATLAB or Simulink related algorithms will error out when using RC-based functions or blocks.

Compatibility Considerations

Conventions for indexing, spatial coordinates, and representation of geometric transforms were changed in R2011b to provide improved interoperability with the Image Processing Toolbox product. Beginning in this release, all Computer Vision System Toolbox blocks, functions, classes, and System objects will only operate in the [x y] coordinate system. Blocks affected by the [x y] coordinate system should be replaced with blocks of the same name from the Vision library. Adjust your models, code, and data as necessary.

For extended details on the coordinate system change, see "Conventions Changed for Indexing, Spatial Coordinates, and Representation of Geometric Transforms" on page 13-2 R2011b Release Notes.

Support for nonpersistent System objects

You can now generate code for local variables that contain references to System objects. In previous releases, you could not generate code for these objects unless they were assigned to persistent variables.

New method for action when System object input size changes

The new `processInputSizeChangeImpl` method allows you to specify actions to take when an input to a System object you defined changes size. If an input changes size after the first call to `step`, the actions defined in `processInputSizeChangeImpl` occur when `step` is next called on that object.

Scaled double data type support for System objects

System objects now support scaled double data types.

Scope Snapshot display of additional scopes in Simulink Report Generator

Using Simulink Report Generator™ software, you can include snapshots of the display produced by a Scope block in a generated report. The Scope Snapshot component, which inserts images of the Simulink Scope block and XY Graph block, now supports the Video Viewer block in Computer Vision System Toolbox software.

Note This feature requires that you have a license for the Simulink Report Generator product.

For more information, see the Simulink Report Generator product documentation.

R2012b

Version: 5.1

New Features

Bug Fixes

Compatibility Considerations

Kalman filter and Hungarian algorithm for multiple object tracking

The `vision.KalmanFilter` object is designed for object tracking. You can use it to predict an object's future location, to reduce noise in the detected location, or to help associate multiple objects with their corresponding tracks. The `configureKalmanFilter` function helps you to set up the Kalman filter object.

The `assignDetectionsToTracks` function assigns detections to tracks in the context of multiple object tracking using the James Munkres' variant of the Hungarian assignment algorithm. The function also determines which tracks are missing, and which detections should begin a new track.

Image and video annotation for detected or tracked objects

The `insertObjectAnnotation` function inserts labels and corresponding circles or rectangles into an image or video to easily display tracked objects. You can use it with either a grayscale or true color image input.

Kanade-Lucas-Tomasi (KLT) point tracker

The `vision.PointTracker` object tracks a set of points using the Kanade-Lucas-Tomasi (KLT), feature tracking algorithm. You can use the point tracker for video stabilization, camera motion estimation, and object tracking.

HOG-based people detector

The `vision.PeopleDetector` object detects people in an input image using the Histogram of Oriented Gradient (HOG) features and a trained Support Vector Machine (SVM) classifier. The object detects unoccluded people in an upright position.

Video file reader support for H.264 codec (MPEG-4) on Windows 7

This release adds H.264 codec (MPEG-4) video formats for Windows 7 operating systems.

Show matched features display

The `showMatchedFeatures` function displays corresponding feature points. It displays a falsecolor overlay of two images with a color-coded plot of the corresponding points connected by a line.

Matching methods added for match features function

This release enhances the `matchFeatures` function for applications in computing the fundamental matrix, stereo vision, registration, and object detection. It provides three different matching methods: simple threshold match, unique matches, and unambiguous matches.

Compatibility Considerations

The new implementation of `matchFeatures` uses different default value for the `method` parameter. If you need the same results as those produced by the previous implementation, set the `Method` parameter with syntax:
matchFeatures(`FEATURES1`, `FEATURES2`, `'Method'`, `'NearestNeighbor_old'`, ...).

Kalman filter for tracking tutorial

The Kalman filter is a popular tool for object tracking. The Using Kalman Filter for Object Tracking example helps you to understand how to setup and use the `vision.KalmanFilter` object and the `configureKalmanFilter` function to track objects.

Motion-based multiple object tracking example

The Motion-Based Multiple Object Tracking example shows you how to perform automatic detection and motion-based tracking of moving objects in a video from a stationary camera.

Face detection and tracking examples

The Face Detection and Tracking example shows you how to automatically detect and a track a face. The Face Detection and Tracking Using the KLT Algorithm example uses

the Kanade-Lucas-Tomasi (KLT) algorithm and shows you how to automatically detect a face and track it using a set of feature points.

Stereo image rectification example

This release enhances the Stereo Image Rectification example. It uses SURF feature detection with the `estimateFundamentalMatrix`, `estimateUncalibratedRectification`, and `detectSURFFeatures` functions to compute the rectification of two uncalibrated images, where the camera intrinsics are unknown.

System object tunable parameter support in code generation

You can change tunable properties in user-defined System objects at any time, regardless of whether the object is locked. For System objects predefined in the software, the object must be locked. In previous releases, you could tune System object properties only for a limited number of predefined System objects in generated code.

save and load for System objects

You can use the `save` method to save System objects to a MAT file. If the object is locked, its state information is saved, also. You can recall and use those saved objects with the `load` method.

You can also create your own `save` and `load` methods for a System object you create. To do so, use the `saveObjectImpl` and `loadObjectImpl`, respectively, in your class definition file.

Save and restore SimState not supported for System objects

The Save and Restore Simulation State as SimState option is no longer supported for any System object in a MATLAB Function block. This option was removed because it prevented parameter tunability for System objects, which is important in code generation.

Compatibility Considerations

If you need to save and restore simulation states, you may be able to use a corresponding Simulink block, instead of a System object.

R2012a

Version: 5.0

New Features

Bug Fixes

Compatibility Considerations

Dependency on DSP System Toolbox and Signal Processing Toolbox Software Removed

The DSP System Toolbox™ and Signal Processing Toolbox™ software are no longer required products for using Computer Vision System Toolbox software. As a result, a few blocks have been modified or removed.

Audio Output Sampling Mode Added to the From Multimedia File Block

The From Multimedia File block now includes a new parameter, which allows you to select frame- or sample-based audio output. If you do not have a DSP System Toolbox license and you set this parameter for frame-based processing, your model will return an error. The Computer Vision System Toolbox software uses only sample-based processing.

Kalman Filter and Variable Selector Blocks Removed from Library

This release removes the Kalman Filter and Variable Selector Blocks from the Computer Vision System Toolbox block library.

Compatibility Considerations

To use these blocks or to run a model containing these blocks, you must have a DSP System Toolbox license.

2-D Median and 2-D Histogram Blocks Replace Former Median and Histogram Blocks

The Median and Histogram blocks have been removed. You can replace these blocks with the 2-D Median and the 2-D Histogram blocks.

Compatibility Considerations

Replace these blocks in your models with the new 2-D blocks from the Computer Vision System Toolbox library.

Removed Sample-based Processing Checkbox from 2-D Maximum, 2-D Minimum, 2-D Variance, and 2-D Standard Deviation Blocks

This release removes the **Treat sample-based row input as a column** checkbox from the 2-D Maximum, 2-D Minimum, 2-D Variance, and 2-D Standard Deviation statistics blocks.

Compatibility Considerations

Modify your code accordingly.

New Viola-Jones Cascade Object Detector

The `vision.CascadeObjectDetector` System object uses the Viola-Jones algorithm to detect objects in an image. This detector includes Haar-like features and a cascade of classifiers. The cascade object detector is pretrained to detect faces, noses and other objects.

New MSER Feature Detector

The `detectMSERFeatures` function detects maximally stable extremal regions (MSER) features in a grayscale image. You can use the `MSERRegions` object, returned by the function, to manipulate and plot MSER features.

New CAMShift Histogram-Based Tracker

The `vision.HistogramBasedTracker` System object uses the continuously adaptive mean shift (CAMShift) algorithm for tracking objects. It uses the histogram of pixel values to identify the object.

New Integral Image Computation and Box Filtering

The `integralKernel` object with the `integralImage` and `integralFilter` functions use integral images for filtering an image with box filters. The speed of the filtering operation is independent of the filter size, making it ideally suited for fast analysis of images at different scales.

New Demo to Detect and Track a Face

This release provides a new demo, Face Detection and Tracking. This example shows you how to develop a simple face tracking system by detecting a face, identifying its facial features, and tracking it.

Improved MATLAB Compiler Support

MATLAB Compiler™ now supports `detectSURFFeatures` and disparity functions.

Code Generation Support

The `vision.HistogramBasedTracker` and `vision.CornerDetector` System objects now support code generation. See About MATLAB Coder for more information about code generation.

Conversion of Error and Warning Message Identifiers

This release changes error and warning message identifiers.

Compatibility Considerations

If you have scripts or functions using message identifiers that have changed, you must update the code to use the new identifiers. Typically, you use message identifiers to turn off specific warning messages. You can also use them in code that uses a try/catch statement and performs an action based on a specific error identifier.

For example, the <'XXXXX:old:ID'> identifier has changed to <'new:similar:ID'>. If your code checks for <'XXXXX:old:ID'>, you must update it to check for <'new:similar:ID'> instead.

To determine the identifier for a warning, run the following command just after you see the warning:

```
[MSG,MSGID] = lastwarn;
```

This command saves the message identifier to the variable `MSGID`.

To determine the identifier for an error that appears at the MATLAB prompt, run the following command just after you see the error.

```
exception = MException.last;
MSGID = exception.identifier;
```

> **Note** Warning messages indicate a potential issue with your code. While you can turn off a warning, a suggested alternative is to change your code without producing a warning.

System Object Updates

Code Generation for System Objects

System objects defined by users now support C code generation. To generate code, you must have the MATLAB Coder™ product.

New System Object Option on File Menu

The File menu on the MATLAB desktop now includes a **New > System object** menu item. This option opens a System object class template, which you can use to define a System object class.

Variable-Size Input Support for System Objects

System objects that you define now support inputs that change size at runtime.

Data Type Support for User-Defined System Objects

System objects that you define now support all MATLAB data types as inputs and outputs.

New Property Attribute to Define States

R2012a adds the new `DiscreteState` attribute for properties in your System object class definition file. Discrete states are values calculated during one step of an object's algorithm that are needed during future steps.

New Methods to Validate Properties and Get States from System Objects

The following methods have been added:

- `validateProperties` – Checks that the System object is in a valid configuration. This applies only to objects that have a defined `validatePropertiesImpl` method
- `getDiscreteState` – Returns a `struct` containing a System object's properties that have the `DiscreteState` attribute

matlab.system.System changed to matlab.System

The base System object class name has changed from `matlab.system.System` to `matlab.System`.

Compatibility Considerations

The previous `matlab.system.System` class will remain valid for existing System objects. When you define new System objects, your class file should inherit from the `matlab.System` class.

R2011b

Version: 4.1

New Features

Bug Fixes

Compatibility Considerations

Conventions Changed for Indexing, Spatial Coordinates, and Representation of Geometric Transforms

Conventions for indexing, spatial coordinates, and representation of geometric transforms have been changed to provide improved interoperability with the Image Processing Toolbox product.

Running your Code with New Conventions

How to run code	Solution
Written with R2011b or later (New User)	You can safely ignore the warning, and turn it off. Your code will use the one-based [x y] coordinate system. To turn the warning off, place the following command in your startup.m file: `warning('off','vision:transition:usesOldCoordinates')`
Written prior to R2011b	To run your pre-R2011b code using the zero-based [row column] conventions, invoke `vision.setCoordinateSystem('RC')` command prior to running your code. Support for the pre-R2011b coordinate system will be removed in a future release. You should update your code to use R2011b coordinate system conventions. To turn the warning off, place the following command in your startup.m file: `warning('off','vision:transition:usesOldCoordinates')`

One-Based Indexing

The change from zero-based to one-based indexing simplifies the ability to blend Image Processing Toolbox functionality with Computer Vision System Toolbox algorithms and visualization functions.

Coordinate System Convention

Image locations in the Computer Vision System Toolbox are now expressed in [x y] coordinates, not in [row column]. The orientation of matrices containing image locations has changed. In previous releases, the orientation was a 2-by-N matrix of zero-based [row column] point coordinates. Effective in R2011b, the orientation is an M-by-2 matrix of

one-based [x y] point coordinates. Rectangular ROI representation changed from [r c height width] to [x y width height].

Example: Convert a point represented in the [r c] coordinate system to a point in the [x y] coordinate system

Convert your data to be consistent with MATLAB and the Image Processing Toolbox coordinate systems by switching the order indexing and adding 1 to each dimension. The *row* index dimension corresponds to the *y* index, and the *column* index corresponds to the *x* index. The following figure shows the equivalent row-column and x-y coordinates for a pixel location in an image.
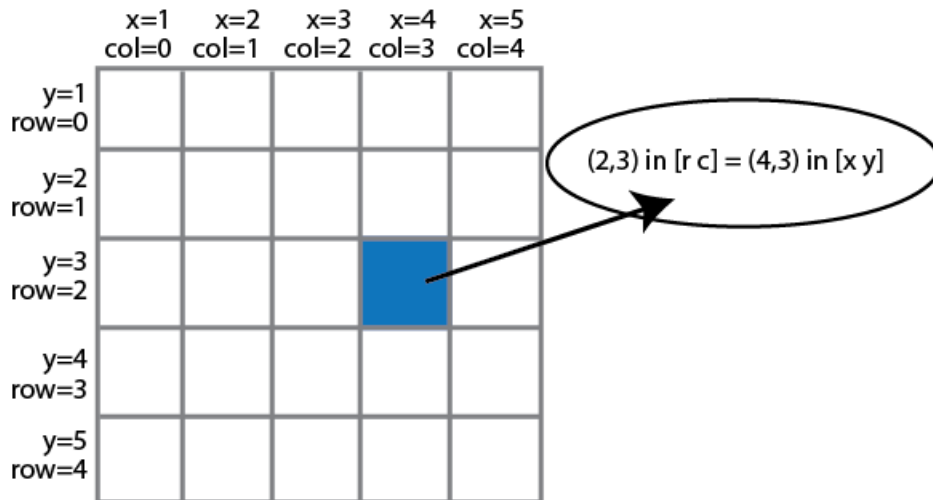
The following MATLAB code converts point coordinates from an [r c] coordinate system to the [x y] coordinate system:

```
ptsRC = [2 0; 3 5]                     % Two RC points at [2 3] and [0 5]
ptsXY = fliplr(ptsRC'+1)     % RC points converted to XY
```

Example: Convert a bounding box represented in the [r c] coordinate system to the [x y] coordinate system

```
% Two bounding boxes represented as [r c height width]
% First box is [2 3 10 5] and the second box is[0 5 15 10]
          bboxRC = [2 0; 3 5; 10 15; 5 10]
% Convert the boxes to XY coordinate system format [x y width heigth]
          bboxXY = [fliplr(bboxRC(1:2,:)'+1) fliplr(bboxRC(3:4,:)')]
```

Example: Convert an affine geometric transformation matrix represented in the [r c] coordinate system to the [x y] coordinate system

```
% Transformation matrix [h1 h2 h3; h4 h5 h6] represented in RC coordinates
          tformRC = [5 2 3; 7 8 13]
% Transformation matrix [h5 h2; h4 h1; h6 h3] represented in XY coordinates
          temp = rot90(tformRC,3);
          tformXY = [flipud(temp(1:2,:)); temp(3,:)]
```

Note: You cannot use this code to remap a projective transformation matrix. You must derive the tformXY matrix from your data.

See Expressing Image Locations for an explanation of pixel and spatial coordinate systems.

Migration to [x y] Coordinate System

By default, all Computer Vision System Toolbox blocks, functions, and System objects are set to operate in the [x y] coordinate system. Use the `vision.setCoordinateSystem` and `vision.getCoordinateSystem` functions to help you migrate your code, by enabling you to revert to the previous coordinate system until you can update your code. Use `vision.setCoordinateSystem('RC')` call to set the coordinate system back to the zero-based [r c] conventions .

For Simulink users, blocks affected by the [x y] coordinate system should be replaced with blocks of the same name from the Vision library. Old blocks are marked with a red "Replace" badge. The following figure shows the Hough Lines block, as it would appear with the Replace badge, indicating that it should be replaced with the Hough Lines block from the R2011b version.
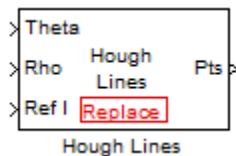
Support for the pre-R2011b coordinate system will be removed in a future release.

Updated Blocks, Functions, and System Objects

The following table provides specifics for the functions, System objects, and blocks that were affected by this update:

Functions	Description of Update	Prior to R2011b	R2011b
`epipolarLine`	The output *A*,*B*,*C* line parameters were changed to work with [x y] one-based coordinates.	$A*\text{row} + B*\text{col} + C$	$A*\text{x} + B*\text{y} + C$
	Accepts Fundamental matrix in [x y] format.		
`estimateFundamentalMatrix`	Adjusted to format of fundamental matrix. Modified to work with points expressed in [x y] one-based coordinates.	[r;c] 2-by-*N* zero-based points.	[x y] *M*-by-2 one-based points.
		Fundamental matrix formatted points for [r;c] zero-based coordinates.	Fundamental matrix formatted to work with [x y] one-based coordinates.
`estimateUncalibratedRectification`	Fundamental matrix, matched points, and output projective transformation matrices provided in new format.	Fundamental matrix formatted only for zero-based [r;c] coordinate system	Fundamental matrix formatted for one-based [x y] coordinate system.
		[r;c] 2-by-*N* zero-based points.	[x y] *M*-by-2 one-based points.
`extractFeatures`	Converted to accept [x y] coordinates	[r;c] 2-by-*N* zero-based points.	[x y] *M*-by-2 one-based points.
`isEpipoleInImage`	Adjusted Fundamental matrix format. Converted to [x y] coordinates.	Fundamental matrix formatted only for zero-based [r;c] coordinate system.	Fundamental matrix formatted only for one-based, [x y] coordinate system.
`lineToBorderPoints`	The input *A*,*B*,*C* line parameters were changed to work with [x y] coordinates.	$A*\text{row} + B*\text{col} + C$, where A, B, and C are represented in a 3-by-*N* matrix of [r;c] zero-based points.	$A*\text{x} + B*\text{y} + C$, where A, B, and C are represented in an *M*-by-3 matrix of [x y] one-based points.

Functions	Description of Update	Prior to R2011b	R2011b
	Output intersection points converted to [x y] one-based coordinate system.	The function returned the intersection points in an 4-by-M matrix. format of [r1;c1;r2;c2] zero-based coordinate system.	The function returns the intersection points in an M-by-4 matrix of format of [x1, y1, x2, y2] one-based coordinate system.
`matchFeatures`	Converted the Index Pairs matrix to match orientation of the POINTS with [x y] one-based coordinates.	The function returns the output Index Pairs in a 2-by-M [r c] zero-based format.	The function returns the output Index Pairs in a M-by-2 [x y] one-based format.
	Changed orientation of input feature vectors.	Input feature vectors stored in columns.	Input feature vectors stored in rows.

System Objects	Description of Update	Prior to R2011b	R2011b
`vision.AlphaBlender`	Converted Location property to take [x y] coordinate location.	Location format in [r;c] zero-based coordinates.	Location format in [x y] one-based coordinates.
`vision.BlobAnalysis`	Centroid and Bounding Box formats converted to [x y] coordinate system.	Centroid format in 2-by-M [r1 r2; c1 c2] zero-based coordinates.	Centroid format in M-by-2 of format [x1 y1 x2 y2] one-based coordinates.
		Bounding Box format in 4-by-N zero-based matrix [r;c;height;width].	Bounding Box format in M-by-4 one-based matrix [x y width height].
`vision.BoundaryTracer`	Converted to accept and output [x y] one-based points.	2-by-N matrix of [r c] zero-based coordinates.	M-by-2 matrix of [x y] one-based coordinates.
`vision.CornerDetector`	Corner locations converted to [x y] coordinate system.	Corner location in a 2-by-N set of [r c] zero-based coordinates.	Corner locations in an M-by-2 one-based [x y] coordinates.
`vision.GeometricScaler`	Converted ROI input to [x y] coordinate one-based system.	Shape in [r c height width] zero-based matrix.	Shape in [x y width height] one-based matrix.

System Objects	Description of Update	Prior to R2011b	R2011b
vision.Geometric Transformer	Converted transformation matrix format to support changed ROI [x y] one-based coordinate system format.	Transformation matrix formatted only for zero-based [r;c] coordinate system.	Takes one-based, [x y] coordinate format for Transformation matrix.
		ROI format in [r;c;height;width] zero-based format.	ROI format in [x y width height] one-based format.
vision.Geometric TransformEstimat or	Converted formatting for input points.	Input points: [r1 r2;c1 c2].	Input points: [x1 y1; x2 y2].
	Converted transformation matrix to [x y] one-based coordinate system.	Transformation matrix formatted only for zero-based [r;c] coordinate system.	Transformation matrix format matches Image Processing Toolbox format.
vision.HoughLine s	Converted format for lines to [x y] one-based coordinate system.	Output: [r11 r21; c11 c21; r12 r22; c12 c22].	Output: [x11 y11 x12 y12; x21 y21 x22 y22].
		Size of output in a 4-by-N zero-based matrix.	Size of the output in M-by-4 one-based matrix.
vision.LocalMaxi maFinder	Converted format for Maxima locations	2-by-N zero-based [r c] coordinates.	M-by-2 one-based [x y] coordinates.
vision.MarkerIns erter	Converted format for locations.	2-by-N zero-based [r c] coordinates.	M-by-2, one-based [x y] coordinates.
vision.Maximum vision.Mean vision.Minimum vision.StandardD eviation vision.Variance	Converted formats for line and rectangle ROIs.	Line: [r1 c1 r2 c2 r3 c3].	Line: [x1 y1 x2 y2 x3 y3].
		Rectangle: [r c height width].	Rectangle: [x y width height].
vision.ShapeInse rter	Converted format for rectangles, lines, polygons, and circles	Rectangle: [r; c; height; width] zero-based format.	Rectangle: [x y width height] one-based format.

System Objects	Description of Update	Prior to R2011b	R2011b
	to [x y] one-based format.	Line: [r1 c1 r2 c2] zero-based format.	Line: [x1 y1 x2 y2] one-based format.
		Polygon: 4-by-M zero-based matrix.	Polygon: M-by-4 one-based matrix.
		Circle: [r c radius] zero-based format.	Circle: [x y radius] one-based format.
	Input image intensity values converted to [x y] one-based format.	N-by-M and N-by-M-by-P [r c] zero-based format.	M-by-N and M-by-N-by-P [x y] one-based format.
vision.TemplateMatcher	Converted Location and ROI format to [x y] one-based coordinate system.	Location output: [r; c] zero-based format.	Location output: [x y] one-based format.
		ROI: [r c height width] zero-based format.	ROI processing: [x y width height] one-based format.
vision.TextInserter	Converted location and color orientation.	2-by-N zero-based [r;c] locations.	M-by-2 [x y] one-based locations.
		$numColorPlanes$-by-N zero-based format.	M-by-$numColorPlanes$ one-based format.
Blocks	**Description of Update**	**Prior to R2011b**	**R2011b**
Apply Geometric Transformation	Converted Transformation matrix format to support changed ROI [x y] one-based coordinate system format.	Transformation matrix formatted only for zero-based [r;c] coordinate system.	Takes one-based, [x y] coordinate format for Transformation matrix.
		ROI format in [r;c;height;width] zero-based format.	ROI format in [x y width height] one-based format.
Blob Analysis	Centroid and Bounding Box formats converted to [x y] coordinate system.	Centroid format in 2-by-M [r1 r2; c1 c2] zero-based coordinates.	Centroid format in M-by-2 of format [x1 y1 x2 y2] one-based coordinates.

Blocks	Description of Update	Prior to R2011b	R2011b
		Bounding Box format in 4-by-N zero-based matrix [r;c;height;width].	Bounding Box format in M-by-4 one-based matrix [x y width height].
Compositing	Converted Location property to takes [x y] coordinate location.	Location format in [r;c] zero-based coordinates.	Location format in [x y] one-based coordinates.
Corner Detection	Corner locations converted to [x y] coordinate system.	Corner location in a 2-by-N set of [r c] zero-based coordinates.	Corner locations in an M-by-2 one-based [x y] coordinates.
Draw Markers	Converted format for locations.	2-by-N zero-based [r c] coordinates.	M-by-2, one-based [x y] coordinates.
Draw Shapes	Converted format for rectangles, lines, polygons, and circles to [x y] one-based format.	Rectangle: [r; c; height; width] zero-based format.	Rectangle: [x y width height] one-based format.
		Line: [r1 c1 r2 c2] zero-based format.	Line: [x1 y1 x2 y2] one-based format.
		Polygon: 4-by-M zero-based matrix.	Polygon: M-by-4 one-based matrix.
		Circle: [r c radius] zero-based format	Circle: [x y radius] one-based format.
Estimate Geometric Transformation	Converted formatting for input points.	Input points: [r1 r2;c1 c2].	Input points: [x1 y1; x2 y2].
	Converted Transformation matrix to [x y] one-based coordinate system.	Transformation: T=[t22 t12 t32; t21 t11 t31; t23 t13 t33].	Transformation matrix format matches Image Processing Toolbox format.
Find Local Maxima	Converted format for Maxima locations	2-by-N zero-based [r c] coordinates.	M-by-2 one-based [x y] coordinates.

Blocks	Description of Update	Prior to R2011b	R2011b
Hough Lines	Converted format for lines to [x y] one-based coordinate system.	Output: [r11 r21; c11 c21; r12 r22; c12 c22].	Output: [x11 y11 x12 y12; x21 y21 x22 y22].
		Size of output in a 4-by-N zero-based matrix.	Size of the output in M-by-4 one-based matrix.
Template Matching	Converted Location and ROI format to [x y] one-based coordinate system.	Location output: [r; c] zero-based format.	Location output: [x y] one-based format.
		ROI: [r c height width] zero-based format.	ROI processing: [x y width height] one-based format.
nsert Text	Converted location and color orientation.	2-by-N zero-based [r;c] locations.	M-by-2 [x y] one-based locations.
		numColorPlanes-by-N zero-based format.	M-by-*numColorPlanes* one-based format.
2-D Maximum2-D Mean2-D Minimum2-D Standard Deviation2-D Variance	Converted formats for line and rectangle ROIs.	Line: [r1 c1 r2 c2 r3 c3].	Line: [x1 y1 x2 y2 x3 y3].
		Rectangle: [r c height width].	Rectangle: [x y width height].
Resize	Converted ROI input to [x y] coordinate one-based system.	Shape in [r c height width] zero-based matrix.	Shape in [x y width height] one-based matrix.
Trace Boundary	Converted to accept and output [x y] one-based points.	2-by-N matrix of [r c] zero-based coordinates.	M-by-2 matrix of [x y] one-based coordinates.

Compatibility Considerations

Blocks affected by the [x y] coordinate system should be replaced with blocks of the same name from the Vision library. Old blocks are marked with a red "Replace" badge. The following figure shows a block which was affected by the coordinate system change:

Hough Lines
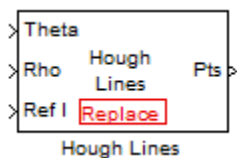
Adjust your model and data as necessary. All functions and System objects are updated to use the one-based [x y] convention.

By default, all Computer Vision System Toolbox blocks, functions, and System objects are set to operate in the [x y] coordinate system. Use the `vision.setCoordinateSystem` and `vision.getCoordinateSystem` functions to help migrate your code containing System objects and functions to the [x y] coordinate system. Use `vision.setCoordinateSystem('RC')` call to temporarily set the coordinate system to old conventions.

When you invoke an affected block, object, or function, a one time, per MATLAB session, warning appears.

See the section, Expressing Image Locations for a description of the coordinate systems now used by the Computer Vision System Toolbox product.

New SURF Feature Detection, Extraction, and Matching Functions

This release introduces a new Speeded Up Robust Features (SURF) detector with functions supporting interest feature detection, extraction and matching. The `detectSURFFeatures` function returns information about SURF features detected in a grayscale image. You can use the `SURFPoints` object returned by the `detectSURFFeatures` function to manipulate and plot SURF features.

New Disparity Function for Depth Map Calculation

The new `disparity` function provides the disparity map between a pair of stereo images. You can use the `disparity` function to find relative depth of the scene for tasks such as, segmentation, robot navigation, or 3-D scene reconstruction.

Added Support for Additional Video File Formats for Non-Windows Platforms

The From Multimedia File block and the `vision.VideoFileReader` now support many compressed video file formats on Linux and Macintosh OS X platforms.

Variable-Size Support for System Objects

Computer Vision System Toolbox System objects support inputs that change their size at run time.

New Demo to Retrieve Rotation and Scale of an Image Using Automated Feature Matching

This release provides a new demo, Finding the Rotation and Scale of an Image Using Automated Feature Matching. This demo shows you how to use the `vision.GeometricTransformEstimator` System object and the new `detectSURFFeatures` function to find the rotation angle and scale factor of a distorted image.

Apply Geometric Transformation Block Replaces Projective Transformation Block

The Projective Transformation block will be removed in a future release. It is recommended that you replace this block with the combination of Apply Geometric Transformation and the Estimate Geometric Transformation blocks to apply projective or affine transform to an image.

Trace Boundaries Block Replaced with Trace Boundary Block

This release provides a replacement block for the Trace Boundaries block. The Trace Boundary block now returns variable size data. See Working with Variable-Size Signals for more information about variable size data.

Note Unlike the Trace Boundaries block, the new Trace Boundary block only traces a single boundary.

The Trace Boundaries block will be removed in a future release.

Compatibility Considerations

The new Trace Boundary block no longer provides the **Count** output port that the older Trace Boundaries block provided. Instead, the new Trace Boundary block and the corresponding `vision.BoundaryTracer` System object now return variable size data.

FFT and IFFT Support for Non-Power-of-Two Transform Length with FFTW Library

The 2-D FFT and 2-D IFFT blocks and the `vision.IFFT` and `vision.FFT` System objects include the use of the FFTW library. The blocks and objects now support non-power-of-two transform lengths.

vision.BlobAnalysis Count and Fill-Related Properties Removed

The blob analysis System object now supports variable-size outputs. Therefore, the `Count` output, and the `NumBlobsOutputPort`, `FillEmptySpaces`, and `FillValues` properties related to fixed-size outputs, were removed from the object.

Compatibility Considerations

Remove these properties from your code, and update accordingly. If you require an explicit blob count, call `size` on one of the object's outputs, such as `AREA`.

vision.CornerDetector Count Output Removed

The corner detector System object now supports variable-size outputs. Therefore, the `Count` output related to fixed-size outputs, were removed from the object.

Compatibility Considerations

Update your code accordingly. If you require an explicit count, call `size` on the object `METRIC` output.

vision.LocalMaximaFinder Count Output and CountDataType Property Removed

The local maxima finder System object now supports variable-size outputs. Therefore, the `Count` output, and the `CountDataType` property related to fixed-size outputs, were removed from the object.

Compatibility Considerations

Remove the property from your code, and update accordingly.

vision.GeometricTransformEstimator Default Properties Changed

The following default property values for the `vision.GeometricTransformEstimator` System object have been changed to provide more reliable outputs.

Property	Default Value	
	From	To
Transform	Projective	Affine
AlgebraicDistanceThreshold	1.5	2.5
PixelDistanceThreshold	1.5	2.5
NumRandomSamplings	100	500
MaximumRandomSamples	200	1000

Compatibility Considerations

The effect of these changes make the object's default-value computations more reliable. If your code relies on the previous default values, you might need to update the affected property values.

Code Generation Support

The `vision.IFFT` System object now supports code generation. See About MATLAB Coder for more information about code generation.

vision.MarkerInserter and vision.ShapeInserter Properties Not Tunable

The following `vision.MarkerInserter` and `vision.ShapeInserter` properties are now nontunable:

- `FillColor`
- `BorderColor`

When objects are locked (for instance, after calling the `step` method), you cannot change any nontunable property values.

Compatibility Considerations

Review any code that changes any `vision.MarkerInserter` or `vision.ShapeInserter` property value after calling the `step` method. You should update the code to use property values that do not change.

Custom System Objects

You can now create custom System objects in MATLAB. This capability allows you to define your own System objects for time-based and data-driven algorithms, I/O, and visualizations. The System object API provides a set of implementation and service methods that you incorporate into your code to implement your algorithm. See Define New System Objects in the DSP System Toolbox documentation for more information.

System Object DataType and CustomDataType Properties Changes

When you set a System object, fixed-point `<xxx>DataType` property to `'Custom'`, it activates a dependent `Custom<xxx>DataType` property. If you set that dependent `Custom<xxx>DataType` property before setting its `<xxx>DataType` property, a warning message displays. `<xxx>` differs for each object.

Compatibility Considerations

Previously, setting the dependent `Custom<xxx>DataType` property would automatically change its `<xxx>DataType` property to `'Custom'`. If you have code that sets the dependent property first, avoid warnings by updating your code. Set the `<xxx>DataType` property to `'Custom'` before setting its `Custom<xxx>DataType` property.

Note If you have a `Custom<xxx>DataType` in your code, but do not explicitly update your code to change `<xxx>DataType` to `'Custom'`, you may see different numerical output.

R2011a

Version: 4.0

New Features

Bug Fixes

Compatibility Considerations

Product Restructuring

The Video and Image Processing Blockset has been renamed to Computer Vision System Toolbox. This product restructuring reflects the broad expansion of computer vision capabilities for the MATLAB and Simulink environments. The Computer Vision System Toolbox software requires the Image Processing Toolbox and DSP System Toolbox software.

You can access archived documentation for the Video and Image Processing Blockset™ products on the MathWorks website.

System Object Name Changes

Package Name Change

The System object package name has changed from video to vision. For example, `video.BlobAnalysis` is now `vision.BlobAnalysis`.

Object Name Changes

The 2D System object names have changed. They no longer have 2D in the name and now use the new package name.

Old Name	New Name
video.Autocorrelator2D	`vision.Autocorrelator`
video.Convolver2D	`vision.Convolver`
video.Crosscorrelator2D	`vision.Crosscorrelator`
video.DCT2D	`vision.DCT`
video.FFT2D	`vision.FFT`
video.Histogram2D	`vision.Histogram`
video.IDCT2D	`vision.IDCT`
video.IFFT2D	`vision.IFFT`
video.MedianFilter2D	`vision.MedianFilter`

New Computer Vision Functions

Extract Features

The `extractFeatures` function extracts feature vectors, also known as descriptors, from an image.

Feature Matching

The `matchFeatures` function takes a pair of feature vectors, as returned by the `extractFeatures` function, and finds the features which are most likely to correspond.

Uncalibrated Stereo Rectification

The `estimateUncalibratedRectification` function returns projective transformations for rectifying stereo images.

Determine if Image Contains Epipole

The `isEpipoleInImage` function determines whether an image contains an epipole. This function supports the `estimateUncalibratedRectification` function.

Epipolar Lines for Stereo Images

The `epipolarLine` computes epipolar lines for stereo images.

Line-to-Border Intersection Points

The `lineToBorderPoints` function calculates the location of the point of intersection of line in an image with the image border. This function supports the `epipolarLine` function.

New Foreground Detector System Object

The `vision.ForegroundDetector` object computes a foreground mask using Gaussian mixture models (GMM).

New Tracking Cars Using Gaussian Mixture Models Demo

The new Tracking Cars Using Gaussian Mixture Models demo illustrates the use of Gaussian mixture models for detection and tracking of cars. The algorithm detects and tracks the cars in a video by separating them from their background.

Expanded To Video Display Block with Additional Video Formats

The To Video Display block now supports 4:2:2 YCbCr video input format.

New Printing Capability for the mplay Function and Video Viewer Block

You can now print the display information from the GUI interface of the mplay function and the Video Viewer block.

Improved Display Updates for mplay Function, Video Viewer Block and vision.VideoPlayer System Object

R2011a introduces the capability to improve the performance of `mplay`, the Video Viewer block and the `vision.VideoPlayer` System object by reducing the frequency with which the display updates. You can now choose between this new enhanced performance mode and the old behavior. By default, all scopes operate in the new enhanced performance mode.

Improved Performance of FFT Implementation with FFTW library

The 2-D FFT, 2-D IFFT blocks include the use of the FFTW library.

Variable Size Data Support

The Resize block now supports variable size data. See Working with Variable-Size Signals for more information about variable size data.

System Object Input and Property Warnings Changed to Errors

When a System object is locked (e.g., after the `step` method has been called), the following situations now produce an error. This change prevents the loss of state information.

- Changing the input data type
- Changing the number of input dimensions
- Changing the input complexity from real to complex
- Changing the data type, dimension, or complexity of tunable property
- Changing the value of a nontunable property

Compatibility Considerations

Previously, the object issued a warning for these situations. The object then unlocked, reset its state information, relocked, and continued processing. To update existing code so that it does not error, use the `release` method before changing any of the items listed above.

System Object Code Generation Support

The following System objects now support code generation:

- `vision.GeometricScaler`
- `vision.ForegroundDetector`

MATLAB Compiler Support for System Objects

The Computer Vision System Toolbox supports the MATLAB Compiler for all objects except `vision.VideoPlayer`. With this capability, you can use the MATLAB Compiler to take MATLAB files, which can include System objects, as input and generate standalone applications.

R2010a MAT Files with System Objects Load Incorrectly

If you saved a System object to a MAT file in R2010a and load that file in R2011a, MATLAB may display a warning that the constructor must preserve the class of the

returned object. This occurs because an aspect of the class definition changed for that object in R2011a. The object's saved property settings may not restore correctly.

Compatibility Considerations

MAT files containing a System object saved in R2010a may not load correctly in R2011a. You should recreate the object with the desired property values and save the MAT file.

Documentation Examples Renamed

In previous releases, the examples used throughout the Video and Image Processing Blockset™ documentation were named with a `doc_` prefix. In R2011a, this changed to a `ex_` prefix. For example, in R2010b, you could launch an example model using the Video Viewer block by typing `doc_thresholding` at the MATLAB command line. To launch the same model in R2011a, you must type `ex_thresholding` at the command line.

Compatibility Considerations

You can no longer launch Video and Image Processing Blockset™ documentation example models using the `doc_` prefix name. To open these models in R2011a, you must replace the `doc_` prefix in the model name with `ex_`.

R2010b

Version: 3.1

New Features

Bug Fixes

Compatibility Considerations

New Estimate Fundamental Matrix Function for Describing Epipolar Geometry

New `Estimate Fundamental Matrix` function for describing epipolar geometry. Epipolar geometry applies to the geometry of stereo vision, where you can calculate depth information based on corresponding points in stereo image pairs. The function supports the generation of embeddable C code.

New Histogram System Object Replaces Histogram2D Object

The new `video.Histogram` System object replaces the video.Histogram2D System object. The name change was made to align this object with its corresponding block.

Compatibility Considerations

The `video.Histogram2D` System object now issues a warning. Update code that uses the 2D-Histogram object to use the new Histogram object.

New System Object release Method Replaces close Method

The `close` method has been replaced by the new `release` method, which unlocks the object and releases memory and other resources, including files, used by the object. The new `release` method includes the functionality of the old `close` method, which only closed files used by the object.

Compatability Considerations

The `close` method now issues a warning. Update code that uses the `close` method to use the new `release` method.

Expanded Embedded MATLAB Support

Embedded MATLAB® now supports the generation of embeddable C code for two Image Processing Toolbox functions and additional Video and Image Processing Blockset System objects. The generated C code meets the strict memory and data type requirements of embedded target environments. Video and Image Processing Blockset provides Embedded MATLAB support for these Image Processing Toolbox functions. See Code Generation for details, including limitations.

Supported Image Processing Toolbox Functions

```
label2rgb
fspecial
```

Supported System objects

Video and Image Processing Blockset objects now support code generation:
```
video.CornerDetector
video.GeometricShearer
video.Histogram
video.MorpologicalBottomHat
video.MorpologicalTopHat
video.MultimediaFileReader
video.MultimediaFileWriter
```

Data Type Assistant and Ability to Specify Design Minimums and Maximums Added to More Fixed-Point Blocks

The following blocks now offer a **Data Type Assistant** to help you specify fixed-point data types on the block mask. Additionally, you can now enable simulation range checking for certain data types on these blocks. To do so, specify appropriate minimum and maximum values on the block dialog box. The blocks that support these features are:

- 2-D DCT
- 2-D FFT
- 2-D IDCT
- 2-D IFFT
- 2-D FIR Filter

For more information on these features, see the following sections in the Simulink documentation:

- Using the Data Type Assistant
- Signal Ranges

Data Types Pane Replaces the Data Type Attributes and Fixed-Point Panes on Fixed-Point Blocks

In previous releases, some fixed-point blocks had a **Data type attributes** pane, and others had a **Fixed-point** pane. The functionality of these panes remains the same, but the pane now appears as the **Data Types** pane on all fixed-point Computer Vision System Toolbox blocks.

Enhanced Fixed-Point and Integer Data Type Support with System Objects

For nonfloating point input, System objects now output the data type you specify. Previously, the output was always a fixed-point, numeric `fi` object.

Compatability Considerations

Update any code that takes nonfloating point input, where you expect the object to output a `fi` object.

Variable Size Data Support

Several Video and Image Processing Blockset blocks now support changes in signal size during simulation. The following blocks support variable size data as of this release:

PSNR	2-D Correlation
Median Filter	2-D Convolution
Block Processing	2-D Autocorrelation
Image Complement	Deinterlacing
Gamma Correction	

See Working with Variable-Size Signals for more information about variable size data.

Limitations Removed from Video and Image Processing Blockset Multimedia Blocks and Objects

Support for reading interleaved AVI data and reading AVI files larger than 2GB on UNIX platforms. Previously, this was only possible on Windows platforms. The following blocks and System objects have the limitation removed:

From Multimedia File block
`video.MultimediaFileReader` System object

Support for writing AVI files larger than 2GB on UNIX platforms, which was previously only possible on Windows platforms. The following blocks and System objects have the limitation removed:
To Multimedia File block
`video.MultimediaFileWriter` System object

R2010a

Version: 3.0

New Features

Bug Fixes

New System Objects Provide Video and Image Processing Algorithms for use in MATLAB

System Objects are algorithms that provide stream processing, fixed-point modeling, and code generation capabilities for use in MATLAB programs. These new objects allow you to use video and image processing algorithms in MATLAB, providing the same parameters, numerics and performance as corresponding Video and Image Processing Blockset blocks. System objects can also be used in Simulink models via the Embedded MATLAB Function block.

Intel Integrated Performance Primitives Library Support Added to 2-D Correlation, 2-D Convolution, and 2-D FIR Filter Blocks

The 2-D Correlation, 2-D Convolution, and 2-D FIR Filter blocks are now taking advantage of SSE Intel instruction set and multi-core processor capabilities for double and single data types.

Variable Size Data Support

Several Video and Image Processing Blockset blocks now support changes in signal size during simulation. The following blocks support variable size data as of this release:

2-D FFT	Hough Transform
2-D FIR Filter	Image Data Type Conversion
Apply Geometric Transformation	Image Pad
Autothreshold	Insert Text
Bottom-hat	Label
Chroma Resampling	2-D Maximum
Closing	2-D Mean
Color Space Conversion	
Compositing	2-D Minimum
Contrast Adjustment	Opening
Dilation	Rotate
Edge Detection	2-D Standard Deviation
Erosion	Template Matching

Estimate Geometric Transformation	To Video Display
Find Local Maxima	Top-hat
Frame Rate Display	2-D Variance
Gaussian Pyramid	Video Viewer

See Working with Variable-Size Signals for more information about variable size data.

Expanded From and To Multimedia File Blocks with Additional Video Formats

The To Multimedia File and From Multimedia File blocks now support 4:2:2 YCbCr video formats.

The To Multimedia File block now supports `WMV`, `WMA`, and `WAV` file formats on Windows platforms. This block now supports broadcasting `WMV` and `WMA` streams over the network.

New Simulink Demos

The Video and Image Processing Blockset contain new and enhanced demos.

New Modeling a Video Processing System for an FPGA Target Demo

This demo uses the Video and Image Processing Blockset in conjunction with Simulink HDL Coder™ to show a design workflow for generating Hardware Design Language (HDL) code suitable for targeting video processing application on an FPGA. The demo reviews how to design a system that can operate on hardware.

New System Object Demos

New Image Rectification Demo

This demo shows how to rectify two uncalibrated images where the camera intrinsics are unknown. Rectification is a useful procedure in many computer vision applications. For example, in stereo vision, it can be used to reduce a 2-D matching problem to a 1-D search. This demo is a prerequisite for the Stereo Vision demo.

New Stereo Vision Demo

This demo computes the depth map between two rectified stereo images using block matching, which is the standard algorithm for high-speed stereo vision in hardware

systems. It further explores dynamic programming to improve accuracy, and image pyramiding to improve speed.

New Video Stabilization Using Point Feature Matching

This demo uses a point feature matching approach for video stabilization, which does not require knowledge of a feature or region of the image to track. The demo automatically searches for the background plane in a video sequence, and uses its observed distortion to correct for camera motion. This demo presents a more advanced algorithm in comparison to the existing Video Stabilization demo in Simulink.

SAD Block Obsoleted

The new Template Matching block introduced in the previous release, supports Sum of Absolute Differences (SAD) algorithm. Consequently, the SAD Block has been obsoleted.