

Image Processing Toolbox™ Release Notes



MATLAB®

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.
1 Apple Hill Drive
Natick, MA 01760-2098

Image Processing Toolbox™ Release Notes

© COPYRIGHT 2000–2022 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

R2022b

Geometric Transformations: Perform geometric transformations using premultiply matrix convention	1-2
Volume Visualization: Enhanced volume, surface, and scene rendering	1-2
EXR Files: Read and write images stored in the EXR file format	1-3
Color Conversion: Support for ProPhoto (ROMM RGB) color space	1-3
DICOM: Support for reading video data	1-3
viscircles Function: Accepts scalar radii	1-3
localapfilt Function: Improved performance	1-3
C Code Generation: Generate code from additional functions using MATLAB Coder	1-4
GPU Code Generation: Generate CUDA code from additional functions using GPU Coder	1-4
GPU Acceleration for gray2ind Function	1-4
Thread-Based Environment: Run functions in a thread-backed pool	1-4
Functionality being removed or changed	1-5
Postmultiply geometric transformation objects are not recommended	1-5
volshow function creates Volume object	1-5
labelvolshow and images.compatibility.volshow.R2022a.volshow will be removed	1-6

R2022a

Deep Learning: Added examples using deep neural networks	2-2
Image Browser App: Additional Import and Export Capabilities	2-2
Image Region Analyzer App: Additional Export Capabilities	2-2

imblatfilt Function: Improved performance for uint8 and single data types	2-2
dicomCollection Function: Improved performance	2-3
C Code Generation: Generate code from additional functions using MATLAB Coder	2-3
GPU Code Generation: Generate CUDA code from additional functions using GPU Coder	2-3
Functionality being removed or changed	2-3
The imsharpen function uses different color space conversion operations for RGB images	2-3
The regionprops function always stores the Image, ConvexImage, and FilledImage properties as cell arrays in the output table for all inputs	2-4

R2021b

Blocked Images: Create and display labeled blocked images	3-2
DICOM: Find and set attributes in DICOM metadata	3-2
Image Quality Metrics: Calculate SSIM metric of deep learning arrays and specify dimensions of computation	3-2
Deep Learning: Added examples using deep neural networks	3-2
medfilt3 Function: Improved performance for small neighborhood sizes	3-2
C Code Generation: Generate code from five functions using MATLAB Coder	3-3
C Code Generation: Generate portable C code that has improved performance for seven functions	3-3
GPU Acceleration for ssim Function	3-4
Thread-Based Environment: Run functions in a thread-backed pool	3-4
Functionality being removed or changed	3-4
The Format property of the GenericImage object is not recommended	3-4

Hyperspectral Image Processing: Image Processing Toolbox Hyperspectral Imaging Library (August 2021, Version 21.1.3)	4-2
Hyperspectral Image Processing: Image Processing Toolbox Hyperspectral Imaging Library (May 2021, Version 21.1.2)	4-2
Deep Learning Networks: Create and combine encoder and decoder modules for networks with repetitive structures	4-3
Deep Learning Networks: Create generators and discriminators for GAN networks	4-3
Deep Learning Data Processing: Rearrange data between dimensions and resize data to match feature map	4-4
Deep Learning: Added examples using deep neural networks	4-4
Blocked Images: Process N-D images that are too large to fit in memory	4-5
Volume Segmenter app supports blocked images	4-5
RAW Files: Read RAW files and color filter array data, and convert to RGB images	4-6
Image Quality Metrics: Calculate metrics of deep learning arrays and specify dimensions of computation	4-6
Image Quality Test Charts: Read and analyze additional versions of Imatest eSFR test charts	4-6
Image Quality Test Charts: Measure scene illuminant using X-Rite ColorChecker test chart	4-6
Registration Estimator App: Support for KAZE and ORB registration techniques	4-7
imerase: Erase rectangular region of image	4-7
randomWindow2d: Select rectangular region of image	4-7
C Code Generation: Generate code from the adapthisteq function using MATLAB Coder	4-7
C Code Generation: Generate portable C code with improved performance for six functions	4-7
GPU Code Generation Support for resize2dLayer Object	4-8

GPU Acceleration for imcrop, multissim, multissim3, psnr, and wiener2 Functions and Enhanced GPU Support for imwarp	4-8
Functionality being removed or changed	4-8
bigimage object and bigimagedatastore object will be removed	4-8
randomCropWindow2d function will be removed	4-8
esfrChart object now refines slanted edge ROI positions	4-8

R2020b

Hyperspectral Image Processing: Image Processing Toolbox Hyperspectral Imaging Library (January 2021, Version 20.2.3)	5-2
Hyperspectral Image Processing: Image Processing Toolbox Hyperspectral Imaging Library (October 2020, Version 20.2.1)	5-3
Volume Segmenter App: Segment 3-D grayscale or RGB volumetric images	5-4
Deep Learning: Resize layers and deep learning arrays	5-4
Deep Learning: Added example using deep neural networks	5-4
Big Images: Select locations of blocks to read	5-4
Image Quality Metrics: Measure image color using X-Rite ColorChecker test chart	5-4
Color Diagrams: Display color measurements from arbitrary number of ROIs and plot measurements in u'v'L color space	5-4
Color Error: Calculate color differences using CIE76, CIE94, or CIEDE2000 standard	5-5
Color Conversion: Support for wide-gamut RGB color spaces	5-5
Geometric Transformations: Perform rigid 2-D and 3-D transformations	5-5
DICOM-RT Contours: Create volumetric mask from contour data	5-5
poly2label Function: Create label matrix from set of ROIs	5-5
tiffreadVolume Function: Read volumetric data from TIFF file	5-5
label2rgb Function Enhancement: Return list of colors	5-6
ROI Objects: Control label transparency, text color, and marker size ...	5-6
multissim and multissim3 Functions: Improved Performance	5-6

C Code Generation: Improved execution speed of generated portable C code for fifteen functions	5-7
Functionality being removed or changed	5-7
The Mask and InclusionThreshold properties of the bigimageDatastore object are not recommended	5-7

R2020a

Hyperspectral Image Processing: Image Processing Toolbox Hyperspectral Imaging Library (July 2020, Version 20.1.1)	6-2
Big Images: Support for class balancing, labeled data, and additional TIFF compression schemes	6-2
Deep Learning: Added example using deep neural networks	6-3
Image Quality Metrics: Measure multi-scale structural similarity (MS-SSIM) index	6-3
modefilt Function: Perform mode filtering	6-3
DICOM-RT Contours: Extract ROI contour data from DICOM-RT structure set	6-3
DICOM Functions: Read and write DICOS file format	6-3
obliqueslice Function: Extract oblique slice from 3-D volume	6-4
makehdr Function Enhancement: Support cell array of matrices as input	6-4
Rectangular ROIs: Control label visibility	6-4
Random Patch Extraction Datastore: Extract random patches from additional types of datastores	6-4
GPU Acceleration for imwarp Function	6-4
DICOM Dictionary Upgrade	6-5
Support for Categorical Data	6-5
C Code Generation: Improved execution speed of generated portable C code for six functions	6-5
Functionality being removed or changed	6-5
im2java2d function will be removed	6-5

Big Images: Process images that are too large to fit in memory	7-2
Deep Learning Data Preprocessing: Perform additional image augmentations	7-2
Random Patch Extraction Datastore: Extract patches from 3-D data and transformed datastores, and train in parallel	7-2
Deep Learning: Added example using deep neural networks	7-2
inpaintExemplar Function: Fill damaged regions in images with exemplar-based inpainting	7-2
DICOM Volume: Construct isotropic volume from DICOM images	7-3
ROI Tools: Create crosshair shape and other enhancements	7-3
View 3-D Volumes as Slice Planes	7-3
imlocalbrighten Function: Brighten dark areas of images	7-5
reducepoly Function: Reduce density of points in ROIs	7-6
Volume Viewer App: Create new session, export visualization settings, and other enhancements	7-6
imshow Function: Specify interpolation method	7-7
volshow Function: Control lighting in volume rendering	7-7
affineOutputView Function: Control view of warped images	7-7
Image Cropping: Crop 3-D volumes and crop using spatial referencing	7-7
Support for Categorical Data	7-7
C Code Generation: Generate code from the imregcorr function using MATLAB Coder	7-7

ROI Creation Functions: New cuboid shape added	8-2
ROI Creation Functions: New bringToFront Function	8-2

Enhanced Volume Display: View labeled volumes and specify colormap	8-2
Deep Learning: Added example using deep neural networks	8-4
Measurements of Region Properties: Measure circularity and Feret properties	8-4
NIFTI File Format Enhancements: Read and write neuroscience image volumes in the NIFTI-2 file format	8-5
Camera Response: Estimate real-world illumination as a function of pixel intensity	8-5
inpaintCoherent Function: Fill damaged regions in images with coherence transport based inpainting	8-5
burstinterpolant Function: Generate a high-resolution image from a burst of lower resolution images	8-5
rgb2lightness Function: Convert an RGB image to a lightness image ...	8-5
Performance Improvements: Performance enhancements for image warping	8-5
C Code Generation: Generate code from four additional functions using MATLAB Coder	8-6
Functionality being removed or changed	8-6
The imshow function now displays large images passively	8-6
The imfindcircles function uses new filter size for logical images	8-6

R2018b

Random Patch Extraction Datastore: Extract random image patches to split up large images for deep learning workflows	9-2
Deep Learning: Added example using deep neural networks	9-2
New Set of ROI Creation Functions	9-2
Volume Show: Visualize 3-D image volumes using the volshow command	9-3
Image Segmentation: Segment 2-D images and N-D volumes using k-means clustering	9-3
Geometric Transformation Objects: Represent and apply custom 2-D and 3-D geometric transformations	9-4

fspecial3: Create predefined 3-D filters	9-4
imflatfield: Perform flat-field correction	9-4
imnlmfilt: Perform non-local means filtering	9-4
imsplit: Split an N-channel image into individual channels	9-4
piqe: Measure image quality using perception-based image quality evaluator (PIQE)	9-4
tonemapfarbman: Reduce dynamic range of HDR images	9-4
fibermetric: Added 3-D support	9-4
Performance improvements: Performance enhancements for 2-D and 3-D morphology, image warping, and fibermetric	9-5
C Code Generation: Generate code from three additional functions using MATLAB Coder	9-5
Functionality being removed or changed	9-5
fibermetric calculates new default structural sensitivity	9-5
Old ROI classes are not recommended	9-5

R2018a

Deep Learning: Added examples using deep neural networks	10-2
Deep Learning Data Preprocessing: Efficiently read and add noise to images for training and prediction	10-2
Image Segmenter: Segment images interactively using new techniques including local graph cut	10-2
Edge-Aware Filtering: Smooth images and reduce noise while preserving edge sharpness with bilateral filtering and anisotropic diffusion filtering	10-2
blendexposure: Perform exposure fusion	10-2
imregmtb: Register images using median threshold bitmaps	10-3
montage: Display multiple images from ImageDatastore object, and specify size of the image thumbnails displayed	10-3
Image Morphology: Perform morphological operations on 3-D volumes, and perform skeletonization on all objects in 2-D image or 3-D binary volume	10-3

Performance: Improved performance in functions including 3-D imwarp, 3-D imfilter, entropyfilt, ordfilt2, medfilt2, and bwmorph	10-3
C Code Generation: Generate code from one additional function using MATLAB Coder	10-3
Functionality being removed or changed	10-4
montage function has several behavior changes	10-4
denoisingImageSource object is removed	10-4
denoisingImageSource function will be removed	10-5

R2017b

Deep Learning: Denoise images using deep learning techniques	11-2
3-D Image Processing: Process 3-D volumetric image data with support for several additional functions	11-2
Image Enhancement: Adjust colors with automatic white balancing, and reduce haze in images	11-2
Image Quality Metrics: Measure image quality without a reference image, and model image quality using an eSFR test chart	11-2
NIFTI File Format: Read and write neuroscience image volumes in the NIFTI file format	11-3
Image Segmenter: Segment images using new techniques including texture segmentation	11-3
Image Segmentation: Calculate similarity coefficients	11-4
DICOM Browsing: Explore the contents of DICOM media in a browser and programmatically	11-4
Image Warper: Transform group of images quickly using the images.geotrans.Warper object	11-5

R2017a

Image Registration App: Explore various registration techniques interactively to align images	12-2
Volume Viewer App: View and slice 3-D volumetric data	12-2

3-D Volumetric Data: Process 3-D volumetric image data with support for over a dozen functions	12-3
fibermetric: Enhance tubular or elongated structures in images	12-3
Image Segmenter App: Added support for RGB images and graph cut segmentation	12-4
lazysnapping: Segmentation technique	12-4
N-D Histograms: Enhance contrast and adjust histograms of N-D images	12-4
dicomread Supports JPEG Variant	12-4
imfilter can return different results for codegen with certain inputs ...	12-4
Functions Being Removed or Changed	12-4

R2016b

Image Browser App: View multiple images and import selected image into apps	13-2
Color Thresholder App: View color data as point cloud for segmentation	13-3
Edge-Aware Filtering: Perform edge-aware filtering based on Laplacian pyramids	13-3
3-D Superpixels: Use simple linear iterative clustering (SLIC) with volumetric images	13-4
3-D Median Filtering: Apply median filter to volumetric image data ...	13-4
colorcloud: Display color information as a point cloud	13-4
edge: Perform faster Canny edge detection	13-4
imshow now sets colormap of axes	13-4
nitfread Supports JPEG2000 Compression	13-6
imregdemons Supports 3-D Images on GPU	13-6
Contrast Adjustment Tool Now Supports Rsets	13-6
Specify Initial Folder Opened in Open Image Dialog Box	13-6
Functions Being Removed	13-6

superpixels: Use simple linear iterative clustering (SLIC) to group pixels for efficient segmentation of color and grayscale images	14-2
Image Segmenter: Segment images using new techniques, including flood-fill and adaptive thresholding	14-2
Image Batch Processor: Export non-image results using expanded workflows	14-2
imgradient3 and imgradientxyz: Calculate 3-D gradient magnitude, direction, and elevation	14-2
C Code Generation: Generate code from 20 additional functions using MATLAB Coder	14-2
imbinarize, otsuthresh, and adaptthresh: Threshold images using global and locally adaptive thresholds	14-3
Structuring Elements: New shapes and a new function	14-3
DICOM functions support non-ASCII character sets	14-3
edge: Change to Canny Edge Detection	14-3
Performance improvements	14-4
Color space conversion functions: improved precision and numerical consistency	14-4
Functions Being Removed	14-4

C Code Generation: Generate code from more than 20 functions using MATLAB Coder	15-2
gabor and imgaborfilt: New class and function for designing and applying Gabor filter banks for use in edge and texture analysis	15-2
grayconnected and imboxfilt: Segment regions by intensities and apply spatial domain filters	15-2
Performance: Performance improvements for grayscale morphology and image filtering	15-2
dpxread and dpxinfo: Read Digital Picture Exchange files	15-3

imwarp function supports new SmoothEdges parameter	15-3
DICOM functions support new options	15-3
New example shows use of imaging functions with the Vision HDL Toolbox	15-3

R2015a

C Code Generation: Generate code from more than 20 additional functions, including regionprops, watershed, bweuler, bwlabel, bwperim, and multithresh, using MATLAB Coder	16-2
App for batch processing sets of images	16-2
Fast geodesic interactive segmentation	16-3
Optimized function for Gaussian filtering	16-3
Fill entire region including border pixels	16-4
Visualize results of boundary tracing	16-4
Examine contents of DICOM files	16-4
Live image capture in Color Thresholder app	16-4
regionprops function can return results in table	16-4
GPU acceleration for imregionalmax, imregionalmin, imgaussfilt, imgaussfilt3, and regionprops functions	16-4
Functions Being Removed	16-5

R2014b

Apps for image segmentation and region analysis	17-2
Image Segmenter app	17-2
Image Region Analyzer app	17-3
C Code Generation: Generate code from 16 additional functions, including bwtraceboundary, imadjust, imclearborder, and medfilt2, using MATLAB Coder	17-4
Nonrigid image registration	17-5

Image warping using displacement fields	17-5
Image segmentation using the Fast Marching Method algorithm	17-5
Image comparison using mean-squared error	17-5
Image filtering based on object properties	17-6
Color space conversion functions	17-6
activecontour function supports parameter to control tendency of contour to expand or contract	17-6
Region-of-Interest (ROI) functions now support deletion from context menu	17-6
dicomwrite function now supports the ability to specify the bitdepth of images written	17-6
GPU acceleration for bwlabel and imregdemons	17-6

R2014a

C Code Generation for more than 25 functions, including edge, imfilter, imwarp, imopen, imclose, imerode, and imdilate using MATLAB Coder	18-2
GPU acceleration for an additional nine functions, including bwdist, imfill, imreconstruct, iradon, radon, and stretchlim	18-2
App for color image thresholding	18-2
Image quality metrics, including peak signal to noise (psnr) and structured similarity metric (ssim)	18-3
Guided filtering for image enhancement	18-3
Phase correlation and translation-only image registration functions ..	18-3
File names of Image Processing Toolbox examples changed	18-4
Location of sample images changed	18-5
regionprops function uses new algorithm to calculate perimeter	18-5

R2013b

GPU acceleration for more than 20 functions, including bwmorph, edge, imresize, and medfilt2	19-2
Additional 2D geometric transformations: piecewise linear, local weighted mean, and polynomial	19-2
Additional parameter in imregister and imregtform to specify initial transformation	19-2
fitgeotrans function for fitting geometric transformation to control point pairs	19-2
imregister and imregtform Return Different Values	19-2
Functions Being Removed	19-3

R2013a

Image segmentation using active contours	20-2
Classes and functions for representing and applying 2-D and 3-D geometric transformations	20-2
Classes for defining world coordinate system of an image	20-2
Code generation for conndef, imcomplement, imfill, imhmax, imhmin, imreconstruct, imregionalmax, imregionalmin, iptcheckconn, and padarray functions (using MATLAB Coder)	20-2
GPU acceleration for imrotate, imfilter, imdilate, imerode, imopen, imclose, imtophat, imbothat, imshow, padarray, and bwlookup functions (using Parallel Computing Toolbox)	20-2
Unsharp mask filtering	20-3

R2012b

Image gradient computation with imgradient and imgradientxy functions	21-2
Histogram matching with imhistmatch function	21-2
Multilevel thresholding with multithresh and imquantize functions ...	21-2

3-D image registration with imregister function	21-2
Code generation for bwmorph and bwlookup with MATLAB Coder	21-2
Added function bwlookup	21-2
Writing private metadata when anonymizing DICOM files	21-2
Expanded color options with imshowpair	21-2
Performance improvements	21-3
New Example	21-3

R2012a

Intensity-Based Image Registration	22-2
Two New Functions to Visually Compare Images	22-2
Circle Detection Using the Circular Hough Transform	22-2
Performance Improvements	22-2
New and Updated Demos	22-2
Data Type Change to Output Variable for bwdist	22-2
iradon Function Updated	22-3
Functions Being Removed	22-3

R2011b

Parallel Block Processing Now Possible with blockproc	23-2
New bwdistgeodesic Function Computes Geodesic Distance Transform	23-2
New graydist Function Computes Gray-Weighted Distance Transform	23-2
New imapplymatrix Function Computes Linear Combination of Color Channels	23-2

Performance Improvements	23-2
Faster Functions	23-2
hdrread Now Corrects for Small Values	23-2
Change in Behavior for dicomwrite	23-2
Warning and Error ID Changes	23-3
Functions and Function Elements Being Removed	23-3

R2011a

New bwconvhull Function Computes Convex Hull Image	24-2
New dicomwrite Option Writes Multiframe Imagery to Single File	24-2
nitfread Now Reads NITF Files with JPEG-Compressed Images	24-2
Reduced Memory Use for std2	24-2
Reduced Memory Use for watershed	24-2
iccread and iccwrite Now Warn in Cases of Unrecognized PrimaryPlatform Signatures	24-3
Plot Selector Now Includes implay	24-3
Support for Code Generation from MATLAB	24-3
edge Function No Longer Smooths Image Twice	24-3
Functions and Function Elements Being Removed	24-3

R2010b

New corner Function Detects Corners in Image	25-2
Efficient Display and Navigation of Very Large Images of Arbitrary Format in imtool	25-2
Now Possible to Control Padding Behavior when Using the blockproc Function	25-2
Writing to JPEG2000 File Format Supported by blockproc	25-2

Enhancements to the dicomread Function	25-2
The ImageMagnification Field of the nitfinfo Function Now Returns a Numeric Value	25-2
Performance Improvements	25-3
Faster Functions	25-3
Functions and Function Elements Being Removed	25-3

R2010a

New ImageAdapter Class Supports Custom File Formats for blockproc	26-2
The blockproc Function Now Supports Spatially Varying Operations ...	26-2
Plot Selector Now Generates Plots for imshow and imtool	26-2
makecform Now Supports White Point Adaptation	26-2
Intel Integrated Performance Primitives Library Support Extended to imdilate, imerode, and medfilt2	26-2
imreconstruct Now Supports int64 and uint64	26-2
Performance Improvements	26-3
Faster Functions	26-3
Multithreaded Functions	26-3
Non-interactive Syntax of improfile Returns Different Output	26-3

R2009b

New blockproc Function to Process Large Images	27-2
Intel Integrated Performance Primitives Library Upgraded and Support Extended to maci64	27-2
Expanded hough Function Allows Specification of Arbitrary Theta Search Space	27-2
The imfilter Function Now Faster for uint16 and double Inputs	27-3
Improved Speed for Calculating N-D Euclidean Distance Transforms with the bwdist Function	27-4

Modified Behavior for the regionprops ConvexHull Property	27-4
Efficient Display and Navigation of Very Large NITF-File Images in imtool	27-4
Performance Improvements	27-4
Faster Functions	27-4
Multithreaded Functions	27-5

R2009a

Faster, Less Memory-Intensive Workflow for Labeling Regions and Measuring Their Properties	28-2
Multithreaded Implementation of imfilter Function	28-2
Efficient Display and Navigation of Very Large Images in imtool	28-2
New Dialog Box for Setting Toolbox Preferences	28-2
New imcolormaptool Function That Opens Choose Colormap Tool	28-2
End Point and Branch Point Detection Now Possible	28-3
nitfread Now Allows Image Subregion Selection	28-3
Support for Intel IPP on Mac	28-3
getColor, getLabelVisible, and setLabelVisible Methods Added to imdistline	28-3
Five Functions Moved to MATLAB	28-3
Fan-Beam Functions Updated	28-4

R2008b

Performance Improvements	29-2
New cornermetric Function Detects Corners	29-2
Now Support Absolute Colorimetric Rendering Intent for GrayTRC and MatTRC	29-2
New createMask Method Creates Mask for Any ROI	29-2

Interactive Tools Refresh when Target Image Changes	29-2
The imscrollpanel 'PreserveView' Parameter Now Works for Images of All Sizes	29-2
Distance Tool and Cropping Tool Now Modes in imtool	29-2
In imtool Opening Adjust Contrast Tool No Longer Selects Window/Level Tool	29-3
immovie Command No Longer Shows Preview	29-3
Replace Calls to ipttable Function with MATLABuitable Function	29-3
imcontour Second Output Argument Changed	29-4
impixelinfo Tool Disappears when Image Changes	29-4
Some Code Moved into Different Directories	29-5
Functions and Demos Being Removed	29-5

R2008a

Create High Dynamic Range (HDR) Images and Write Them to Files ..	30-2
Measure Properties of Regions in Grayscale Images	30-2
Display Very Large Images by Subsampling	30-2
Enhancements to ROI Tools	30-2
ROI Tools Reimplemented as MATLAB Classes	30-2
ROI Tools Support New wait and resume Methods	30-2
Interactively Add New Vertices to ROI Polygons	30-3
Enhancements to Color Functions	30-3
makecform Supports Converting Between sRGB and CMYK	30-3
iccwrite Creates Smaller ICC Profiles	30-3
cp2tform Function Supports New Transformations	30-3
hough Function Uses Specified RhoResolution Values	30-3
Enhancements to Interactive Tools	30-3
New and Updated Demos	30-4
Enhancements to Other Functions	30-4

New Interactive Image Sequence and Video Viewer	31-2
Image Tool Includes Cropping, Enhanced Contrast Adjustment, and Saving of Modified Images	31-2
New Function for Converting Bayer Pattern Encoded Images to RGB	31-2
New Function for Creating a Multiresolution Gaussian Pyramid	31-2
Enhanced ROI Definition Behavior for imcrop, roifill, and roipoly	31-2
New Modular Interactive GUI-building Tools	31-3
New Programmable ROI Tools	31-3
Support for Reading NITF and HDR Images	31-3
Enhanced Performance	31-3
DICOM Dictionary Upgrade	31-3
Changes to Other Functions	31-4

Enhancements to imresize Function	32-2
applycform Supports Tetrahedral Interpolation	32-2
Control Point Selection Tool Enhancements	32-2
Enhancements to impoint, imline, and imrect Functions	32-2
Enhancements to montage Function	32-2
makecform Uses 'icc' Whitepoint for L*a*b*/sRGB Conversions	32-3
normxcorr2 Might Return Different Results	32-3
watershed Uses New Algorithm	32-3
Changes to Other Functions	32-3

R2006b

Enhancements to DICOM Capabilities	33-2
New Symmetric Option with graycomatrix Function	33-2
Enhancements to ICC Color Capabilities	33-2
Enhancements to the imdistline Function	33-2
setColor Method Accepts Predefined Color Strings	33-3

R2006a

Enhanced ICC Profile Capabilities	34-2
New Pointer Management Functions	34-2
New Constraint Creation Function	34-2
Functions cp2tform, tforminv, imtransform	34-2
IPPL Not Used on 64-Bit Systems	34-2

R14SP3

Support for Two New Medical Image File Formats	35-2
New Point, Rectangle, and Line Functions	35-2
Image Tool Enhancements and Improvements	35-2
New Distance Tool	35-2
Adjust Contrast Tool Enhancements and Improvements	35-2
New Utility Functions for Use with Profile-Based Color Space Conversion Functions	35-2
New Documentation on Processing Image Sequences	35-2
Control Point Selection Tool Now Works on Macintosh Systems	35-2
Obsolete and Deleted Functions	35-2
Image Tool is Not Compilable	35-3

Major Bug Fixes	36-2
Major Revisions to Fan-Beam Functions	36-2
Changes to the DICOM Functions	36-2
Changes to Image Tool and Modular Interactive Tools	36-3
Changes to the imshow Function	36-4
Fixes to Other Functions	36-4
Fixes to Image Processing Toolbox Deployment Issues	36-4
 New Directory Needed	 36-4

R2022b

Version: 11.6

New Features

Bug Fixes

Compatibility Considerations

Geometric Transformations: Perform geometric transformations using premultiply matrix convention

Starting in R2022b, Image Processing Toolbox functions use a premultiply convention to create and perform geometric transformations.

A new set of objects enables geometric transformations using a premultiply convention.

Transformation Type	2-D Object	3-D Object
Affine transformation	<code>affinetform2d</code>	<code>affinetform3d</code>
Rigid transformation	<code>rigidtfom2d</code>	<code>rigidtfom3d</code>
Similarity transformation	<code>simmtform2d</code>	<code>simmtform3d</code>
Translation transformation	<code>transltform2d</code>	<code>transltform3d</code>
Projective transformation	<code>projtform2d</code>	Not applicable

The new `fitgeotform2d` function estimates a 2-D geometric transformation that maps pairs of control points between two images.

Several functions are updated to use the premultiply convention. These functions accept the new geometric transformation objects as input, or return the objects as output:

- `affineOutputView`
- `imregcorr`
- `imregister`
- `imregtform`
- `imwarp`
- `randomAffine2d`
- `randomAffine3d`
- `Warper`

Compatibility Considerations

The old geometric transformation objects that use a postmultiply convention, such as `affine2d`, are not recommended. You can streamline your geometric transformation workflows by switching to the new objects that use the premultiply convention. For more information, see [Compatibility Considerations](#) on page 1-5.

Volume Visualization: Enhanced volume, surface, and scene rendering

New functions and objects offer enhanced visual quality and interactive performance, as compared to functions and objects in previous releases, when displaying and interacting with 3-D volumes and surfaces.

- A `Viewer3D` object controls the appearance and behavior of a 3-D scene. Create a `Viewer3D` object using the `viewer3d` function.
- A `Volume` object controls the appearance and behavior of a volume within a 3-D scene. Create a `Volume` object using the `volshow` function.

-
- A Surface object controls the appearance and behavior of a 3-D surface within a 3-D scene.

Compatibility Considerations

The `volshow` function now creates a Volume object instead of a `volshow` object. For more information, see Compatibility Considerations on page 1-5.

EXR Files: Read and write images stored in the EXR file format

The EXR format stores image data according to the OpenEXR file specification. To work with EXR files, use these functions:

- `isexr` — Check whether a file is a valid EXR file.
- `exrinfo` — Read information about image data from EXR files.
- `exrread` — Read image data from EXR files.
- `exrwrite` — Write image data to EXR file.
- `exrHalfASingle` — Convert numeric values into half-precision values.

Color Conversion: Support for ProPhoto (ROMM RGB) color space

These functions now support the ProPhoto color space: `lin2rgb`, `rgb2lin`, `lab2rgb`, `rgb2lab`, `rgb2xyz`, `xyz2rgb`, and `chromadapt`. To use the ProPhoto color space, specify the `ColorSpace` name-value argument as "prophoto-rgb". The ProPhoto color space is also known as the Reference Output Medium Metric (ROMM) RGB color space, and has a larger gamut than the sRGB and Adobe 1998 RGB color spaces.

DICOM: Support for reading video data

The `dicomread` function now supports reading DICOM files that contain video data with MPEG-2, MPEG-4/H.264 and HEVC/H.265 encoding. Because MATLAB® decodes video data using the codecs installed on your computer, some files might not be able to be read on all platforms. For more details, see "Supported Video and Audio File Formats".

viscircles Function: Accepts scalar radii

`viscircles` accepts a scalar value for the `radii` argument. When `radii` is a scalar, `viscircles` draws all circles with the same radius.

localapfilt Function: Improved performance

The `localapfilt` function shows improved performance. For example, this code is about 6.8x faster than in the previous release.

```
function localapfiltTimingTest
    A = imread("peppers.png");
    localapfilt(A,0.4,0.5);
end
```

The approximate execution times are:

R2022a: 0.254 s

R2022b: 0.037 s

The code was timed on a Windows® 10, Intel® Xeon® E5-2683 v4 CPU @ 2.10 GHz test system (two processors) using the `timeit` function:

```
timeit(@locallapfiltTimingTest)
```

C Code Generation: Generate code from additional functions using MATLAB Coder

The list indicates the Image Processing Toolbox functions that have been enabled for code generation in this release. For all target platforms, these functions generate C code. For a complete list of Image Processing Toolbox functions that support code generation, see [Functions Supporting Code Generation](#).

- `bwpropfilt`
- `iradon`
- `psf2otf`
- `radon`

The `edge` function now supports code generation for the directional gradient output arguments, `Gx` and `Gh`.

GPU Code Generation: Generate CUDA code from additional functions using GPU Coder

The list indicates the Image Processing Toolbox functions that have been enabled for optimized CUDA® code generation in this release.

- `adaptthresh`
- `histeq`
- `label2rgb`

GPU Acceleration for `gray2ind` Function

R2022b adds GPU acceleration for the `gray2ind` function. GPU acceleration requires a Parallel Computing Toolbox™ license and meeting the GPU computing requirements detailed [here](#).

Thread-Based Environment: Run functions in a thread-backed pool

R2022b adds thread-based support for the Image Processing Toolbox functions listed in the table. Run code in the background on a single thread using MATLAB `backgroundPool`, or accelerate code with multiple thread workers using `ThreadPool`. Use of multiple thread workers requires Parallel Computing Toolbox.

<code>adapthisteq</code>	<code>bfscore</code>	<code>bwconncomp</code>	<code>bwdist</code>
<code>bwferet</code>	<code>bwlabel</code>	<code>bwlabeln</code>	<code>bwlookup</code>

bwmorph3	bwpack	bwskel	bwtraceboundary
bwunpack	demosaic	edge	edge3
entropyfilt	exrHalfAsSingle	exrinfo	exrread
exrwrite	fibermetric	grabcut	histeq
hough	imabsdiff	imapplymatrix	imbothat
imboxfilt	imboxfilt3	imclose	imdiffusefilt
imdilate	imerode	imfill	imfilter
imgaussfilt	imgaussfilt3	imgradient	imgradient3
imhist	imlincomb	imnlmfilt	imopen
imreconstruct	imregionalmax	imregionalmin	imsegfmm
imtophat	inpaintCoherent	inpaintExemplar	integralImage
integralImage3	intlut	iradon	isexr
label2idx	medfilt2	medfilt3	modefilt
multissim	multissim3	nitfread	obliquesslice
ordfilt2	planar2raw	poly2mask	qtdecomp
radon	raw2planar	raw2rgb	rawread
regionprops	regionprops3	rgb2lab	rgb2lightness
superpixels	superpixels3	watershed	

Functionality being removed or changed

Postmultiply geometric transformation objects are not recommended

Still runs

These geometric transformation objects are not recommended because they use the postmultiply matrix convention: `affine2d`, `affine3d`, `rigid2d`, `rigid3d`, and `projective2d`. There are no plans to remove the old geometric transformation objects at this time. However, you can streamline your geometric transformation workflows by switching to new objects that use the premultiply convention. For more information, see “Migrate Geometric Transformations to Premultiply Convention”.

volshow function creates Volume object

Behavior change

The `volshow` function now creates a `Volume` object instead of a `volshow` object. The `Volume` object offers more rendering styles and integrates with a `Viewer3D` object to offer easier control of the volume visualization. The `Volume` object also supports web graphics.

The `volshow` function accepts a different set of name-value arguments based on the properties of the `Volume` object. For a list of these properties, see `Volume`.

When you call `volshow` without specifying a parent object, the function now creates a new `Viewer3D` object and sets that object as the parent. Before, the function identified the current figure using the `gcf` function and set that figure as the parent.

If you want to reproduce the prior behavior, then use the `images.compatibility.volshow.R2022a.volshow` function to create a `volshow` object.

labelvolshow and images.compatibility.volshow.R2022a.volshow will be removed

Still runs

The `labelvolshow` object and the `images.compatibility.volshow.R2022a.volshow` function will be removed in a future release. Use the `viewer3d` and `volshow` functions instead. The `viewer3d` function creates a `Viewer3D` object that you can use to modify the scene, such as the camera, background colors, and lighting. The `volshow` object creates a `Volume` object that you can use to modify the appearance of the volumetric data, such as the rendering style, colormap, and transparency map. Specify and change the labels and intensity data by setting the `OverlayData` and `Data` properties of the `Volume` object, respectively.

Some of the properties of the `labelvolshow` object and the `volshow` object created using the `images.compatibility.volshow.R2022a.volshow` function have a different name with the `Viewer3D` and `Volume` objects. In particular, use the overlay properties, of the `Volume` object, such as `OverlayColormap` and `OverlayAlphamap`, to adjust the appearance of the volume labels. For more information, see `Viewer3D` and `Volume`.

Discouraged Usage	Recommended Replacement
<p>This example displays volume data using the <code>images.compatibility.volshow.R2022a.volshow</code> function.</p> <pre>vol = images.compatibility.volshow.R2022a.volshow(V);</pre>	<p>Here is equivalent code that displays volume data in a <code>Volume</code> object using the <code>volshow</code> function.</p> <pre>vol = volshow(V);</pre>
<p>This example displays a volume on a blue background using an isosurface rendering style.</p> <pre>vol = images.compatibility.volshow.R2022a.volshow(V, ... BackgroundColor="b",Renderer="Isosurface");</pre> <p>Change the background color and the rendering style by setting properties of the object.</p> <pre>vol.BackgroundColor = "g"; vol.Renderer = "VolumeRendering");</pre>	<p>Here is equivalent code that sets the background color using a <code>Viewer3D</code> object and the rendering style using a <code>Volume</code> object.</p> <pre>viewer = viewer3d(BackgroundColor="b"); vol = volshow(V,Parent=viewer,RenderingStyle="Isosurface");</pre> <p>Change the background color by setting properties of the <code>Viewer3D</code> object and change the rendering style by setting properties of the <code>Volume</code> object.</p> <pre>viewer.BackgroundColor = "g"; vol.RenderingStyle = "VolumeRendering");</pre>
<p>This example displays labeled volume data using the <code>labelvolshow</code> function.</p> <pre>vol = labelvolshow(labels,volume);</pre>	<p>Here is equivalent code that displays labeled volume data in a <code>Volume</code> object using the <code>volshow</code> function.</p> <pre>vol = volshow(volume,OverlayData=labels);</pre>

Discouraged Usage	Recommended Replacement
<p>This example displays a labeled volume on a blue background and specifies the label colors.</p> <pre data-bbox="240 384 833 443">vol = labelvolshow(labels,volume, ... BackgroundColor="b",LabelColor=cmap);</pre> <p>Change the background color and label colors by setting properties of the <code>labelvolshow</code> object.</p> <pre data-bbox="240 556 617 615">vol.BackgroundColor = "g"; vol.LabelColor = cmap2;</pre>	<p>Here is equivalent code that sets the background color using a <code>Viewer3D</code> object and the label colors using a <code>Volume</code> object.</p> <pre data-bbox="857 415 1438 506">viewer = viewer3d(BackgroundColor="b"); vol = volshow(V,OverlayData=labels, ... OverlayColormap=cmap,Parent=viewer);</pre> <p>Change the background color by setting properties of the <code>Viewer3D</code> object and change the label colors by setting properties of the <code>Volume</code> object.</p> <pre data-bbox="857 680 1279 739">viewer.BackgroundColor = "g"; vol.OverlayColormap = cmap2;</pre>

R2022a

Version: 11.5

New Features

Bug Fixes

Compatibility Considerations

Deep Learning: Added examples using deep neural networks

These examples show how to solve image processing problems by using deep neural networks. These examples require Deep Learning Toolbox™.

- The Detect Image Anomalies Using Explainable One-Class Classification Neural Network example has been updated to use a new data set. The new version shows how to train an anomaly detector for visual inspection of pill images.
- The Detect Image Anomalies Using Pretrained ResNet-18 Feature Embeddings example shows how to train a similarity-based anomaly detector using one-class learning of feature embeddings extracted from a pretrained ResNet-18 convolutional neural network.
- The Classify Defects on Wafer Maps Using Deep Learning example shows how to classify eight types of defects on silicon wafer maps using a simple convolutional neural network (CNN).

Image Browser App: Additional Import and Export Capabilities

The **Image Browser** app supports a greater set of options for curating a collection of images:

- Add images in directories or datastores to an already opened collection of images
- Remove a subset of images from a collection
- Export a subset of images to an image datastore
- Export images from a datastore to another datastore

Image Region Analyzer App: Additional Export Capabilities

The **Image Region Analyzer** app supports a greater set of export options:

- Export region properties as a structure or table
- Export a function that enables you to perform identical filtering and measurements on other binary images

imbilatfilt Function: Improved performance for uint8 and single data types

The `imbilatfilt` function shows improved performance for inputs of data type `uint8` and `single`.

For example, this code is about 2.5x faster than in the previous release.

```
function imbilatfiltTimingTest
    im = imread("cameraman.tif");
    im4k = imresize(im,[3840 2160]);
    tic
    out = imbilatfilt(im4k);
    toc
end
```

The approximate execution times are:

R2021b: 0.007 s

R2022a: 0.003 s

The code was timed on a Windows 10, Intel Xeon E5-2683 v4 CPU @ 2.10 GHz test system (two processors) by calling the function `imilatfiltTimingTest`.

dicomCollection Function: Improved performance

The performance of the `dicomCollection` function has been improved. The function is about 30-50% faster than in the previous release.

C Code Generation: Generate code from additional functions using MATLAB Coder

The list indicates the Image Processing Toolbox functions that have been enabled for code generation in this release. For all target platforms, these functions generate C code. For a complete list of Image Processing Toolbox functions that support code generation, see [Functions Supporting Code Generation](#).

- `graythresh`
- `imguidedfilter`
- `imreducehaze`
- `imsharpen`
- `normxcorr2`
- `stdfilt`
- `wiener2`

GPU Code Generation: Generate CUDA code from additional functions using GPU Coder

The list indicates the Image Processing Toolbox functions that have been enabled for optimized CUDA code generation in this release.

- `adapthisteq`
- `imfindcircles`
- `mat2gray`
- `regionfill`

Functionality being removed or changed

The `imsharpen` function uses different color space conversion operations for RGB images *Behavior change*

Starting in R2022a, the `imsharpen` function uses different color space conversion operations to sharpen RGB images. In R2021b and earlier, the `imsharpen` function performed color space conversions using the `makecform` and `applycform` functions. Starting in R2022a, the `imsharpen` function performs color space conversions using the `rgb2lab` and `lab2rgb` functions.

The new operations yield different results for sharpened RGB images. If you need to reproduce the old behavior, then you can replace the call to `imsharpen` with a call to the

`images.compatibility.imsharpen.r2021b.imsharpen` function instead. You do not need to change the input arguments.

The `regionprops` function always stores the `Image`, `ConvexImage`, and `FilledImage` properties as cell arrays in the output table for all inputs

Behavior change

Starting in R2022a, when you specify a table output format, the `regionprops` function stores the `Image`, `ConvexImage`, and `FilledImage` property values as cell arrays, regardless of the size of the image objects. In previous releases, if the size of the bounding box of an object was 1-by-1 or 1-by-*n*, these properties were stored in the output table as a numeric scalar or row vector, respectively.

To update your code, access the value of the `Image`, `ConvexImage`, and `FilledImage` properties using dot notation with curly braces, `{}`. For example, use this code to access the `Image` property for the first object in the input image `BW`. In previous releases, curly braces were not required to access values stored as a numeric scalar or row vector.

```
stats = regionprops("table",BW,"Image");  
imdata = stats.Image{1};
```

R2021b

Version: 11.4

New Features

Bug Fixes

Compatibility Considerations

Blocked Images: Create and display labeled blocked images

The `polyToBlockedImage` function creates a labeled `blockedImage` from a set of coordinates that specify the vertices of one or many ROIs.

The `showLabels` function shows the label data in a `blockedImage` over the image data in a `bigimageshow` object. The `hideLabels` function hides the labels.

DICOM: Find and set attributes in DICOM metadata

The `dicomfind` function finds the location and value of a target attribute in DICOM metadata. The `dicomupdate` function enables you to set the value of a target attribute using the location of the attribute. These functions support nested attributes.

Image Quality Metrics: Calculate SSIM metric of deep learning arrays and specify dimensions of computation

The `ssim` function now accepts `dArray` input for deep learning applications.

This function also supports formatted data with dimension labels of 'S' (spatial), 'C' (channel), and 'B' (batch). The function returns a separate result for each index along the channel and batch dimensions.

Deep Learning: Added examples using deep neural networks

These examples show how to solve image processing problems by using deep neural networks. These examples require Deep Learning Toolbox.

- The Detect Image Anomalies Using Explainable One-Class Classification Neural Network example shows how to detect anomalies such as cracks in concrete using single-class classification.
- Two examples show how to generate a high-quality X-ray computed tomography (CT) image from a noisy CT image. The Unsupervised Medical Image Denoising Using CycleGAN example uses a cycle-consistent generative adversarial network (CycleGAN) and trains on patches of image data from a large collection of chest scans. The Unsupervised Medical Image Denoising Using UNIT example uses a UNIT network and trains using full images from a single chest scan.
- The Recover Images from Extreme Low-Light Conditions Using Deep Learning example shows how to recover brightened RGB images from RAW camera data collected in extreme low-light conditions using a U-Net.
- The Preprocess Multiresolution Images for Training Classification Network example shows how to create datastores that read and preprocess multiresolution whole slide images (WSIs) for the purpose of training a classification network. The Classify Tumors in Multiresolution Blocked Images example shows how to use the datastores to train and evaluate the performance of an Inception-v3 deep neural network.

medfilt3 Function: Improved performance for small neighborhood sizes

The `medfilt3` function shows improved performance for neighborhood sizes from [3, 3, 3] up to [31, 31, 31].

For example, this code is about 3x faster than in the previous release.

```
function timingTestMedfilt3
    load mrystack;
    noisyV = imnoise(mrystack,'salt & pepper',0.2);
    tic
    filteredV = medfilt3(noisyV);
    toc
end
```

The approximate execution times are:

R2021a: 0.24 s

R2021b: 0.08 s

The code was timed on a Windows 10, Intel Xeon Gold 5220 CPU @ 2.2 GHz test system by calling the function `timingTestMedfilt3`.

C Code Generation: Generate code from five functions using MATLAB Coder

The list indicates the Image Processing Toolbox functions that have been enabled for code generation in this release. For all target platforms, this function generates C code. For a complete list of Image Processing Toolbox functions that support code generation, see [Functions Supporting Code Generation](#).

- `imgradient`¹
- `imgradientxy`¹
- `imnoise`
- `poly2mask`
- `regionfill`

¹ If you choose the generic MATLAB Host Computer option in the MATLAB Coder™ configuration settings, this function generates C code that uses a precompiled, platform-specific shared library. Using a shared library preserves performance optimizations in this function but limits the target to only those platforms that support MATLAB (see system requirements).

C Code Generation: Generate portable C code that has improved performance for seven functions

You can generate portable C code that has faster execution speed than in previous releases for these functions:

- `histeq`
- `imadjust`
- `imfill`
- `imfilter`
- `imreconstruct`
- `label2rgb`

- `medfilt2`

The optimizations include multithreading and algorithm improvements. Generating portable C code requires MATLAB Coder.

GPU Acceleration for `ssim` Function

R2021b adds GPU acceleration for the `ssim` function. GPU acceleration requires Parallel Computing Toolbox software and meeting the GPU computing requirements detailed here.

Thread-Based Environment: Run functions in a thread-backed pool

R2021b adds thread-based support for the Image Processing Toolbox functions listed in the table. Run code in the background on a single thread using MATLAB `backgroundPool` or accelerate code with multiple thread workers using `ThreadPool`. Use of multiple thread workers requires Parallel Computing Toolbox.

<code>adaptthresh</code>	<code>affine2d</code>	<code>affine3d</code>	<code>affineOutputView</code>
<code>cpcorr</code>	<code>cpstruct2pairs</code>	<code>findbounds</code>	<code>fitgeotrans</code>
<code>fliptform</code>	<code>gray2ind</code>	<code>grayslice</code>	<code>im2int16</code>
<code>im2single</code>	<code>im2uint16</code>	<code>im2uint8</code>	<code>imcrop</code>
<code>imcrop3</code>	<code>impyramid</code>	<code>imquantize</code>	<code>imref2d</code>
<code>imref3d</code>	<code>imregconfig</code>	<code>imregcorr</code>	<code>imregdemons</code>
<code>imregmtb</code>	<code>imresize3</code>	<code>imrotate</code>	<code>imrotate3</code>
<code>imsplit</code>	<code>imtranslate</code>	<code>imwarp</code>	<code>ind2gray</code>
<code>label2rgb</code>	<code>makesampler</code>	<code>maketform</code>	<code>mat2gray</code>
<code>MattesMutualInformation</code>	<code>MeanSquares</code>	<code>normxcorr2</code>	<code>OnePlusOneEvolutionary</code>
<code>otsuthresh</code>	<code>projective2d</code>	<code>RegularStepGradientDescent</code>	<code>tformarray</code>
<code>tformfwd</code>	<code>tforminv</code>		

Functionality being removed or changed

The `Format` property of the `GenericImage` object is not recommended

Behavior change

Starting in R2021b, the `Format` property of the `GenericImage` object is not recommended. Use the `Extension` property instead. If you specify the `Format` property using dot notation, then the value is assigned to the `Extension` property.

R2021a

Version: 11.3

New Features

Bug Fixes

Compatibility Considerations

Hyperspectral Image Processing: Image Processing Toolbox Hyperspectral Imaging Library (August 2021, Version 21.1.3)

Image Processing Toolbox Hyperspectral Imaging Library now supports singleton dimensions in hyperspectral data cubes.

To use 1-D or 2-D spectral data as the hypercube input to the supported hyperspectral functions, you must reshape it to 3-D volume data:

- Reshape 1-D spectral data of size 1-by- P into a 3-D hypercube of size 1-by-1-by- P .
- Reshape 2-D spectral data of size M -by- P into a 3-D hypercube of either size M -by-1-by- P or 1-by- M -by- P .

These functions in Image Processing Toolbox Hyperspectral Imaging Library support singleton dimensions for hyperspectral data.

Explore, Analyze, and Visualize	Data Correction	Dimensionality Reduction	Spectral Unmixing	Spectral Matching and Target Detection
hypercube	dn2radiance	hyperpca	ppi	sam
	dn2reflectance	hypermnf	fippi	sid
	radiance2Reflectance	inverseProjection	nfindr	jmsam
	empiricalLine		estimateAbundanceLS	sidsam
	flatField			ns3
	iarr			spectralMatch
	logResiduals			ndvi
	subtractDarkPixel			anomalyRX

To use these functions and tools, you must download Image Processing Toolbox Hyperspectral Imaging Library from the Add-On Explorer. See Get and Manage Add-Ons.

Hyperspectral Image Processing: Image Processing Toolbox Hyperspectral Imaging Library (May 2021, Version 21.1.2)

This release introduces new features for smile detection and advanced smile correction. It also adds support for block processing of large-sized hyperspectral data cubes.

- Use the `smileMetric` function to detect spectral smile effect in a hyperspectral data cube.
- The `reduceSmile` function now supports the maximum noise fraction (MNF) transform-based method for smile correction. To select a smile correction method, specify the name-value argument 'Method' with a value of either 'SpectralSmoothing' or 'MNF' for the moving average or MNF method, respectively.

-
- To process a large-sized hyperspectral data cube without running out of memory, you can use block processing. Enable block processing for the functions that support it by specifying the 'BlockSize' name-value argument.

For example, to enable block processing while using the `correct00B` function, specify the name-value argument 'BlockSize', [20 20].

```
newhcube = correct00B(hcube,spectralResponse,'BlockSize',[20 20]);
```

These functions in the Image Processing Toolbox Hyperspectral Imaging Library support block processing.

- `correct00B`
- `dn2radiance`
- `dn2reflectance`
- `radiance2Reflectance`
- `subtractDarkPixel`
- `reduceSmile`
- `spectralIndices`
- `ndvi`

To perform block processing by specifying the 'BlockSize' name-value argument, you must have MATLAB R2021a or a later release.

To use these functions and tools, you must download Image Processing Toolbox Hyperspectral Imaging Library from the Add-On Explorer. See [Get and Manage Add-Ons](#).

Deep Learning Networks: Create and combine encoder and decoder modules for networks with repetitive structures

Create a network with repetitive blocks of layers, such as an encoder module that performs a series of downsampling operations or a decoder module that performs a series of upsampling operations, using the `blockedNetwork` function.

Create an encoder module directly from a pretrained network, such as VGG-19 or Inception-v3, using the `pretrainedEncoderNetwork` function.

Connect an encoder module and a decoder module with an optional bridge, final network, and skip connections using the `encoderDecoderNetwork` function.

These functions require Deep Learning Toolbox.

Deep Learning Networks: Create generators and discriminators for GAN networks

You can create popular GAN generator and discriminator networks: CycleGAN, PatchGAN, pix2pixHD, and UNIT. You can optionally modify the networks by changing aspects such as the number of downsampling operations and the type of activation and normalization operations. The table describes the functions that enable you to create and modify GAN networks.

Network	Creation and Modification Functions
CycleGAN generator network	Create a CycleGAN generator network using the <code>cycleGANGenerator</code> function.
pix2pixHD generator network	Create a pix2pixHD generator network using the <code>pix2pixHDGlobalGenerator</code> . Add a local enhancer to a pix2pixHD generator network using the <code>addPix2PixHDLocalEnhancer</code> function.
UNIT generator network	Create an unsupervised image-to-image translation (UNIT) generator network using the <code>unitGenerator</code> function. Perform image-to-image translation on a trained UNIT network using the <code>unitPredict</code> function.
PatchGAN discriminator network	Create a PatchGAN discriminator network or a pixel discriminator network using the <code>patchGANDiscriminator</code> function.

These functions require Deep Learning Toolbox.

Deep Learning Data Processing: Rearrange data between dimensions and resize data to match feature map

The `DepthToSpace2DLayer` object and `depthToSpace` function rearrange data from the depth dimension to spatial dimensions.

The `SpaceToDepthLayer` object, which rearranges spatial blocks of data into the depth dimension, no longer requires Computer Vision Toolbox™ as of R2021a. The new `spaceToDepth` function also rearranges spatial blocks of data into the depth dimension.

The `resize2dLayer` and `resize3dLayer` objects now support resizing input to the size of a second input feature map connected to the layer. To use this resizing mode, specify the `EnableReferenceInput` property of the layer as `true`.

These functions require Deep Learning Toolbox.

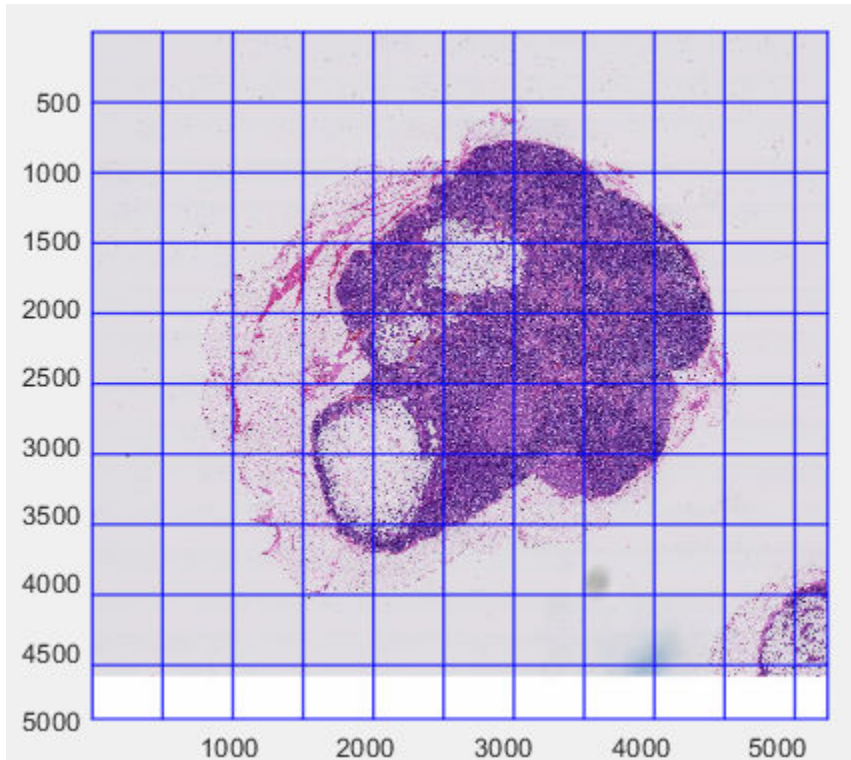
Deep Learning: Added examples using deep neural networks

These examples show how to solve image processing problems by using deep neural networks. These examples require Deep Learning Toolbox.

- The Unsupervised Day-To-Dusk Image Translation Using UNIT example shows how to perform image-to-image translation of images captured during daytime and dusk conditions using a UNIT network.
- The Quantify Image Quality Using Neural Image Assessment example shows how to quantify image quality using the no-reference NIMA model.
- The Classify Large Multiresolution Images Using `blockedImage` and Deep Learning example now demonstrates the new recommended workflow using `blockedImage` and `blockedImageDatastore` objects.

Blocked Images: Process N-D images that are too large to fit in memory

Read and process N-D images that are too large to fit in memory by using a `blockedImage` object. A `blockedImage` object supports images with one or multiple resolution levels. This illustration shows a large, multiresolution image, converted to a blocked image, displayed with block boundaries visible. Each block is 512-by-512.



To manage a collection of image blocks that belong to one or more `blockedImage` objects, use a `blockedImageDatastore` object.

The `blockedImage` and `blockedImageDatastore` objects replace the `bigimage` and `bigimageDatastore` objects. For more information, see [Compatibility Considerations](#) on page 4-8.

Volume Segmenter app supports blocked images

The **Volume Segmenter** app now accepts `blockedImage` objects as input for segmentation. Use blocked images when a volume is too large to fit into memory. By using blocked images, you can segment volumes without running out of memory.

After loading the blocked image into **Volume Segmenter**, you can use the same set of tools the app offers for nonblocked images to segment regions of the volume. This includes the use of automated methods, working block-by-block through each slice. Use the **Blocked Image** tab in **Volume Segmenter** to specify the block you want to work on, and to mark that block completed. **Volume Segmenter** automatically calculates the percent of blocks that are done. **Volume Segmenter**

combines your work across blocks to create masks. For more information, see [Work with Blocked Images Using the Volume Segmenter](#).

RAW Files: Read RAW files and color filter array data, and convert to RGB images

You can read and work with images in the RAW file format. The RAW file format preserves image data in its most unedited state, recorded directly from the camera sensor. Most camera manufacturers define their own proprietary RAW file format, such as the Nikon NEF file format and the Canon® CRW format. Adobe® has also defined a RAW file format, DNG (digital negative), which is supported by several cameras. To work with RAW files, use these functions:

- `rawinfo` - Read information about color filter array (CFA) images in RAW files
- `rawread` - Read CFA images from RAW files
- `raw2rgb` - Convert a RAW file into an RGB file
- `raw2planar` - Separate a Bayer-patterned CFA image into individual, sensor-element images
- `planar2raw` - Combine planar sensor images into a full Bayer-pattern CFA image

Image Quality Metrics: Calculate metrics of deep learning arrays and specify dimensions of computation

The `psnr`, `multissim`, and `multissim3` functions now accept `darray` input for deep learning applications.

These functions also support formatted data with dimension labels of 'S' (spatial), 'C' (channel), and 'B' (batch). For data with a batch dimension, the functions return a separate result for each index along the batch dimension.

Image Quality Test Charts: Read and analyze additional versions of Imatest eSFR test charts

The `esfrChart` object can detect and analyze the Enhanced, Wedge Enhanced, and Wedge Extended versions of the Imatest® eSFR test chart.

You can specify camera parameters to compensate for distortion using the 'CameraParameters' name-value argument.

By default, the `esfrChart` object now refines the position of slanted-edge ROIs based on localized image content. For more information, see [Compatibility Considerations](#) on page 4-8.

Image Quality Test Charts: Measure scene illuminant using X-Rite ColorChecker test chart

You can now measure the scene illuminant of an image of an X-Rite® ColorChecker® test chart by using the `measureIlluminant` function.

Registration Estimator App: Support for KAZE and ORB registration techniques

The **Registration Estimator** app now supports two additional registration techniques: KAZE and ORB.

The KAZE technique is a multiscale 2-D feature detection and description algorithm in nonlinear scale spaces.

The ORB technique detects corners in images with changes in scale, rotations, or both.

imerase: Erase rectangular region of image

Erase a rectangular region from an image using the `imerase` function. You can specify the fill value of the erased region.

randomWindow2d: Select rectangular region of image

Select a rectangular region of an image using the `randomWindow2d` function. You can specify the dimensions of the region or the aspect ratio and area of the region as a fraction of the total image area.

C Code Generation: Generate code from the `adapthisteq` function using MATLAB Coder

You can generate C code from the `adapthisteq` function by using MATLAB Coder. If you choose the generic MATLAB Host Computer target platform, the function generates code that uses a precompiled, platform-specific shared library.

C Code Generation: Generate portable C code with improved performance for six functions

For these functions, you can generate portable C code that has faster execution speed than in previous releases.

- `bwareaopen`
- `bwboundaries`
- `bwlabel`
- `houghpeaks`
- `imbinatfilt`
- `otsuthresh`

The optimizations include the use of multithreading, data parallelization, and SIMD code generation. Generating portable C code requires MATLAB Coder.

GPU Code Generation Support for `resize2dLayer` Object

You can generate optimized CUDA code from the `resize2dLayer` layer. The generated code includes CUDA kernels for parallelizable parts of your deep learning algorithms. To generate this code, you must have a GPU Coder™ license.

GPU Acceleration for `imcrop`, `multissim`, `multissim3`, `psnr`, and `wiener2` Functions and Enhanced GPU Support for `imwarp`

R2021a adds GPU acceleration for the `imcrop`, `multissim`, `multissim3`, `psnr`, and `wiener2` functions. GPU acceleration requires Parallel Computing Toolbox software and meeting the GPU computing requirements detailed here.

The `imwarp` function, which already supports GPU acceleration, now extends that support to include the use of displacement fields.

Functionality being removed or changed

`bigimage` object and `bigimagedatastore` object will be removed

Still runs

The `bigimage` object and the `bigimagedatastore` object will be removed in a future release. To work with large images, use the `blockedImage` object and the `blockedImageDatastore` object instead.

The new blocked image technology offers these advantages:

- Extension to N-D processing
- Introduction of a documented adapter interface to enable custom support for any data source that can be chunked into blocks
- Simpler interface because of use of direct pixel subscripts (`bigimage` uses world coordinates)
- Simpler interface for single-resolution level images (default resolution level is 1)

To display blocked image data, use the `bigimageshow` function.

`randomCropWindow2d` function will be removed

Still runs

The `randomCropWindow2d` function will be removed in a future release. To select random crop windows from an image, use the `randomWindow2d` function instead.

To update your code, change instances of the function name `randomCropWindow2d` to `randomWindow2d`. You do not need to change the input arguments.

`esfrChart` object now refines slanted edge ROI positions

Behavior change

Starting in R2021a, the `esfrChart` object refines the position of slanted edge ROIs based on localized image content by default. In previous releases, the `esfrChart` object did not refine the position of the slanted edge ROIs after the initial estimate of the position. The refinement results in more precise ROI positions.

To replicate the previous behavior, specify the 'RefinePoints' name-value argument as `false` when creating an `esfrChart` object.

R2020b

Version: 11.2

New Features

Bug Fixes

Compatibility Considerations

Hyperspectral Image Processing: Image Processing Toolbox Hyperspectral Imaging Library (January 2021, Version 20.2.3)

This release adds functions for atmospheric correction, hyperspectral image enhancement, and spectral matching. It also adds an example for hyperspectral image classification using a deep learning network.

- Perform atmospheric correction by using the `rrs`, `correct00B`, or `fastInScene` function.
 - `rrs` — Measure remote sensing reflectance from water leaving radiance and solar irradiance values.
 - `correct00B` — Remove out of band effects from multispectral data by using sensor spectral response.
 - `fastInScene` — Remove atmospheric effects by using a fast in-scene method.
- Perform hyperspectral image resolution enhancement by using the `sharpencnmf` function. The function uses the coupled nonnegative matrix factorization (CNMF) method to sharpen a low spatial resolution hyperspectral image by fusing information from high spatial resolution multispectral or panchromatic image.
- Reduce noise in hyperspectral images by using `denoiseNGMeet` function. The function implements a non-local meets global paradigm that exploits spatial non-local similarity and spectral low-rank property for denoising.
- Find spectral match between two spectra and identify unknown spectral signatures by using hybrid spectral matching methods. These methods are combinations of two different distance measures.

Function	Method	Description
<code>sidsam</code>	Spectral information divergence-spectral angle mapper (SID-SAM)	Combines spectral information divergence (SID) and spectral angle mapper (SAM) methods
<code>jmsam</code>	Jeffries Matusita -spectral angle mapper (JMSAM)	Combines Jeffries Matusita (JM) distance and SAM methods
<code>ns3</code>	Normalized spectral similarity score (NS3)	Combines Euclidean distance and SAM methods

You can now use the `spectralMatch` function for matching spectral signatures by using any of these hybrid methods: SID-SAM, JMSAM, and NS3. Set the value of the name-value pair argument 'Method' to

- 'sidsam' for SID-SAM method
- 'jmsam' for JMSAM method
- 'ns3' for NS3 method
- The Classify Hyperspectral Images Using Deep Learning example shows how to classify regions in a hyperspectral image by using a custom spectral convolution neural network (CSCNN) classification network.

To use these functions and tools, you must download Image Processing Toolbox Hyperspectral Imaging Library from the Add-On Explorer. See Get and Manage Add-Ons.

Hyperspectral Image Processing: Image Processing Toolbox Hyperspectral Imaging Library (October 2020, Version 20.2.1)

This release adds enhancements for the hypercube object as well as functions for radiometric calibration, atmospheric correction, smile reduction, and computing multiple spectral indices. It also adds an interactive NDVI thresholding example.

- The hypercube object enhancements include:
 - Read Hyperion Level 1R (Hyperion L1R) satellite data stored in hierarchical data format (HDF). The `hypercube` object can read files with file extension `.L1R`.
 - Create a `hypercube` object with custom metadata by using the syntax

```
hypercube(dataCube,wavelength,metadata)
```

- Perform radiometric calibration of hyperspectral data by using these functions.

Function	Method
<code>dn2radiance</code>	Convert digital numbers to top of the atmosphere (TOA) radiance values.
<code>dn2reflectance</code>	Convert digital numbers to TOA reflectance values.
<code>radiance2Reflectance</code>	Convert TOA radiance values to TOA reflectance values.

- Remove spectral smile effect from hyperspectral data by using the `reduceSmile` function.
- Perform atmospheric correction by using these functions.

Function	Method
<code>empiricalLine</code>	Empirical line calibration
<code>flatField</code>	Flat field correction
<code>iarr</code>	Internal average relative reflectance
<code>logResiduals</code>	Log residuals correction
<code>sharc</code>	Satellite hypercube atmospheric rapid correction
<code>subtractDarkPixel</code>	Dark pixel subtraction

- Find the spectral index values for various materials in a hyperspectral data by using the `spectralIndices` function. You can use the function with either a single supported spectral index or multiple supported spectral indices.
- The Identify Vegetation Regions Using Interactive NDVI Thresholding example shows how to identify different types of vegetation regions in a hyperspectral image based on the normalized vegetation index (NDVI). In this example you interactively set the threshold ranges for identifying vegetation regions in the NDVI image.

To use these functions and tools, you must download Image Processing Toolbox Hyperspectral Imaging Library from the Add-On Explorer. See [Get and Manage Add-Ons](#).

Volume Segmenter App: Segment 3-D grayscale or RGB volumetric images

The **Volume Segmenter** app enables you to create and refine a binary or semantic segmentation mask for a 3-D grayscale or 3-D RGB image using automated, semi-automated, and manual techniques.

Deep Learning: Resize layers and deep learning arrays

The `resize2dLayer` and `resize3dLayer` objects enable you to resize 2-D and 3-D input by a scale factor or to a specified size. The `dlresize` function enables you to resize the spatial dimensions of a `dlarray` object by a scale factor or to a specified size.

Deep Learning: Added example using deep neural networks

The [Develop Raw Camera Processing Pipeline Using Deep Learning](#) example shows how to convert RAW camera data to an aesthetically pleasing color image using a U-Net. This example requires Deep Learning Toolbox.

Big Images: Select locations of blocks to read

The `selectBlockLocations` function selects the location of unique blocks from one or more `bigimage` objects. You can now create a `blockLocationSet` object by using the `selectBlockLocations` function.

The `bigimageDatastore` object no longer supports a mask input. To select block locations within a mask, use the `selectBlockLocations` function. For more information, see [Compatibility Considerations](#) on page 5-7.

Image Quality Metrics: Measure image color using X-Rite ColorChecker test chart

The `colorChecker` object automatically identifies the color patch regions of interest (ROIs) of a ColorChecker test chart produced by X-Rite (formerly GretagMacbeth®). You can display the ROIs overlaid on the chart by using the `displayChart` function. You can measure the colors of an image of the ColorChecker test chart by using the `measureColor` function.

Color Diagrams: Display color measurements from arbitrary number of ROIs and plot measurements in u'v'L color space

The `displayColorPatch` and `plotChromaticity` functions now enable you to visually compare the measured and reference colors of a `colorChecker` object. Likewise, you can use the functions to display measured and reference colors of a chart with a custom number of color ROIs. To display color measurements, specify the `colorTable` input argument as a color table with one row per color ROI.

The `plotChromaticity` function now enables you to plot measured and reference colors in the u'v'L color space. The function also enables you to display the color space as a 3-D color solid.

Color Error: Calculate color differences using CIE76, CIE94, or CIEDE2000 standard

The `deltaE` function calculates color differences between images in the RGB or L*a*b* color space using the CIE76 standard. The `imcolordiff` function calculates color differences between images using the CIE94 or CIEDE2000 standard. You can use name-value pair arguments to adjust perceptual uniformity for different applications.

Color Conversion: Support for wide-gamut RGB color spaces

Modern displays, such as ultra-high definition (UHD) and high dynamic range (HDR) televisions, are capable of reproducing images at a high luminance and wider color gamut than earlier devices. Usage of wide-gamut color spaces, such as BT.2020 and BT.2100, have become more prevalent in color processing. To convert between a wide-gamut RGB color space and the YCbCr or CIE 1931 XYZ color space, use the new conversion functions: `rgbwide2ycbcr`, `ycbcr2rgbwide`, `xyz2rgbwide`, and `ycbcr2rgbwide`.

Geometric Transformations: Perform rigid 2-D and 3-D transformations

The `rigid2d` object defines a 2-D rigid geometric transformation consisting of rotation and translation. The `rigid3d` object, which defines a 3-D rigid geometric transformation and was released in R2020a, no longer requires Computer Vision Toolbox.

Apply the 2-D and 3-D rigid transformations to images by using the `imwarp` function.

DICOM-RT Contours: Create volumetric mask from contour data

The `createMask` function creates a volumetric mask from the contour data in a Digital Imaging and Communication in Medicine radiation therapy (DICOM-RT) structure set.

poly2label Function: Create label matrix from set of ROIs

The `poly2label` function creates a label matrix from a set of ROIs.

tiffreadVolume Function: Read volumetric data from TIFF file

A single TIFF file can contain volumetric data stored as a stack of 2-D slices, each slice in a separate Image File Directory (IFD). Previously, loading this data from the file into the workspace required calling the `imread` function multiple times in a loop. Now, you can read an entire 3-D volume from a TIFF file with one call to the `tiffreadVolume` function. You can also read subsets of the data in the volume using `tiffreadVolume` with the `'pixelregion'` parameter.

label2rgb Function Enhancement: Return list of colors

The `label2rgb` function has an added input argument that enables you to specify the output format of the RGB data. You can use this argument to return a list of colors instead of an RGB image.

ROI Objects: Control label transparency, text color, and marker size

The ROI objects, such as `Circle`, `Ellipse`, and `Rectangle`, support two new properties to control the transparency of label backgrounds and the color of label text, `'LabelAlpha'` and `'LabelTextColor'`. In addition, the ROI objects, except `Cuboid` and `Crosshair`, support a property to control the size of the markers used when drawing an ROI, `'MarkerSize'`. The markers appear at vertices when drawing polygons and other shapes, and as resizing handles for circles and ellipses.

multissim and multissim3 Functions: Improved Performance

The image quality metric functions `multissim` and `multissim3` show improved performance.

For example, this code for the `multissim` function with an image of size 1280-by-780 pixels is about 3x faster than in the previous release.

```
function timingTestMultissim
    Iref = imread('pout.tif');
    Iref = imresize(Iref,[1280 720]);
    I = imnoise(Iref,'salt & pepper',0.05);
    t = timeit(@()multissim(I,Iref));
end
```

For an image of size 1280-by-780 pixels, the approximate execution times are:

R2020a: 0.06 s

R2020b: 0.02 s

This code for the `multissim3` function with a volumetric image of size 128-by-128-by-27 voxels is about 3x faster than in the previous release.

```
function timingTestMultissim3
    load mri;
    Vref = squeeze(D);
    V = imnoise(Vref,'salt & pepper',0.05);
    t = timeit(@()multissim3(V,Vref));
end
```

For a volumetric image of size 128-by-128-by-27 voxels, the approximate execution times are:

R2020a: 0.06 s

R2020b: 0.02 s

The code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.6 GHz test system by calling the functions `timingTestMultissim` and `timingTestMultissim3`.

C Code Generation: Improved execution speed of generated portable C code for fifteen functions

For these functions, you can generate portable C code that has faster execution speed than in previous releases.

- `edge`
- `hough`
- `houghlines`
- `imclose`
- `imdilate`
- `imerode`
- `imhist`
- `imopen`
- `imresize`
- `imwarp`
- `medfilt2`
- `multithresh`
- `regionprops`
- `rgb2ycbcr`
- `ycbcr2rgb`

The optimizations include the use of multithreading, data parallelization, and SIMD code generation. Generating portable C code requires MATLAB Coder.

Functionality being removed or changed

The Mask and InclusionThreshold properties of the bigimageDatastore object are not recommended

Behavior change

Starting in R2020b, the Mask and InclusionThreshold properties of the bigimageDatastore object are not recommended. To select block locations within a mask, use the selectBlockLocations function.

Functionality	Result	Use Instead	Compatibility Considerations
The Mask and InclusionThreshold properties of bigimageDatastore	Still runs	selectBlockLocations	Specify a mask and inclusion threshold as input arguments to the selectBlockLocations function. Specify the output, a blockLocationSet object, as an input to the bigimageDatastore object.

R2020a

Version: 11.1

New Features

Bug Fixes

Compatibility Considerations

Hyperspectral Image Processing: Image Processing Toolbox Hyperspectral Imaging Library (July 2020, Version 20.1.1)

Image Processing Toolbox Hyperspectral Imaging Library offers functions and tools to import, export, process, and visualize hyperspectral images.

- Use the **Hyperspectral Viewer** app to interactively read, visualize, and analyze the spectral bands in a hyperspectral data cube.
- You can read hyperspectral data from various file formats, perform dimensionality reduction, extract endmembers, estimate abundance maps, match spectral signatures, and detect anomalies by using the functions listed in this table.

Read, Write, and Visualize Hyperspectral Data	Select Bands and ROIs	Reduce Spectral Dimensionality
hypercube	selectBands	hyperpca
enviinfo	removeBands	hypermnf
enviwrite	assignData	inverseProjection
colorize	cropData	
Extract Endmembers and Estimate Abundance Maps	Match Spectral Signatures	Measure Vegetation and Detect Anomalies
countEndmembersHFC	readEcostressSig	ndvi
ppi	sam	anomalyRX
fippi	sid	
nfindr	spectralMatch	
estimateAbundanceLS		

- The Hyperspectral Image Analysis Using Maximum Abundance Classification example shows how to identify different regions in an hyperspectral image by using maximum abundance maps.
- The Classify Hyperspectral Image Using Library Signatures and SAM example shows how to classify regions in an hyperspectral image by using reference signatures from ECOSTRESS spectral library and spectral angle mapper technique.
- The Endmember Material Identification Using Spectral Library example shows how to identify the class of endmembers in an hyperspectral image by using ECOSTRESS spectral library signatures.
- The Target Detection Using Spectral Signature Matching example shows how to detect and segment desired targets in an hyperspectral image by using spectral matching method.

To use these functions and tools, you must download Image Processing Toolbox Hyperspectral Imaging Library from the Add-On Explorer. See [Get and Manage Add-Ons](#).

Big Images: Support for class balancing, labeled data, and additional TIFF compression schemes

Workflows involving images that are too large to fit in memory now include these enhancements.

-
- To support class balancing in semantic segmentation and object detection workflows, the `blockLocationSet` object stores the location of blocks to read from a set of big images. To create a `bigimageDatastore` object that reads balanced data, specify a `blockLocationSet` object as an input argument.
 - To support semantic segmentation workflows, the `bigimage` and `bigimageDatastore` objects support categorical label images. You can create a `bigimage` object from categorical data by specifying the `Classes` and `PixelLabelIDs` properties. To calculate the number of pixel labels for each class, use the `countEachLabel` function. To display categorical big image data, use the `bigimageshow` function.
 - To support deep learning workflows, the `apply` function accepts additional input arguments. To pad partial blocks, specify the 'PadPartialBlocks' name-value pair. To include spatial information when processing blocks, specify the 'IncludeBlockInfo' name-value pair. To process multiple blocks at once, specify the 'BatchSize' name-value pair.
 - You can write TIFF files containing big image data by using additional compression schemes. To specify the compression scheme, use the `compression` input argument of the `write` function.

Deep Learning: Added example using deep neural networks

The Neural Style Transfer Using Deep Learning example shows how to apply the stylistic appearance of one image to the scene content of a second image using a modified VGG-19 network.

Image Quality Metrics: Measure multi-scale structural similarity (MS-SSIM) index

The `multissim` and `multissim3` functions calculate the MS-SSIM index for 2-D images and 3-D volumes. The MS-SSIM index measures the structural similarity of the images at varying scales, which can be more robust to variations in viewing conditions.

modefilt Function: Perform mode filtering

The `modefilt` function performs mode filtering on a 2-D image or 3-D volume. Using optional name-value pairs, you can specify the size of the filter and padding options. `modefilt` can be particularly useful when filtering categorical or labeled data.

DICOM-RT Contours: Extract ROI contour data from DICOM-RT structure set

The `dicomContours` function extracts region of interest (ROI) contour data from Digital Imaging and Communication in Medicine radiation therapy (DICOM-RT) structure set files. The `dicomContours` function returns the extracted ROI information as a `dicomContours` object. You can add, delete, modify, and display ROI contour data by using the functions `addContour`, `deleteContour`, `convertToInfo`, and `plotContour`, respectively.

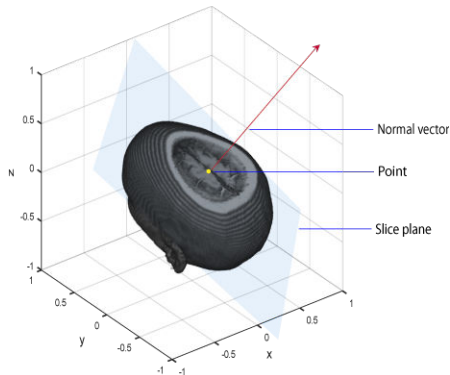
DICOM Functions: Read and write DICOS file format

The `dicomread`, `dicomwrite`, and `dicominfo` functions can now read and write data from files in the Digital Imaging and Communications in Security (DICOS) format. These functions already support

the Digital Imaging and Communications in Medicine (DICOM) format. DICOS is a file format used in applications such as airport security screening.

obliquelice Function: Extract oblique slice from 3-D volume

The `obliquelice` function extracts 2-D oblique slices from 3-D volumetric data with reference to a given point in the volume and a normal vector. The slicing plane is perpendicular to the normal vector and passes through the specified point.



makehdr Function Enhancement: Support cell array of matrices as input

The `makehdr` function now supports a cell array of matrices as an input. The function can now generate a single-precision, high dynamic range image from the set of spatially registered, low dynamic range image matrices stored in the input cell array.

Rectangular ROIs: Control label visibility

The `LabelVisible` property of the `Rectangle` ROI object supports the value `'inside'`. When you specify `'inside'`, the ROI displays a label only when the ROI contains enough room for the label to fit inside the ROI. The `drawrectangle` function also supports this value for its `LabelVisible` name-value pair argument.

Random Patch Extraction Datastore: Extract random patches from additional types of datastores

When you create a `randomPatchExtractionDatastore`, you can now specify the input data as a `TransformedDatastore` containing any type of underlying datastore.

GPU Acceleration for imwarp Function

This release adds GPU acceleration for the `imwarp` function. GPU acceleration requires Parallel Computing Toolbox software and meeting the GPU computing requirements detailed here.

DICOM Dictionary Upgrade

The default DICOM dictionary has been upgraded to the 2019 version released by NEMA. A text version of this dictionary is included in the product, `dicom-dict.txt`.

Compatibility Considerations

If your DICOM code depends on hard-coded attribute names, you may see failures because some of the names have changed. In addition, some DICOM files may no longer parse. Customers who require attribute settings from the 2007 version can use the `dicomdict` function to access the old data dictionary, which is included in R2020a. That is, `dicom-dict.txt` will have 2019 values and `dicom-dict-2007.txt` is the version of `dicom-dict.txt` found in R2007b through R2019b.

Support for Categorical Data

These functions now accept and return categorical image data: `imrotate`, `imrotate3`, `imtranslate`, `label2rgb`, `padarray`, `regionprops`, and `regionprops3`.

The `imwarp` function now enables you to specify any valid category as a fill value.

C Code Generation: Improved execution speed of generated portable C code for six functions

For these functions, you can generate portable C code that has faster execution speed than in previous releases.

- `edge`
- `imfilter`
- `imrotate`
- `imwarp`
- `medfilt2`
- `multithresh`

The optimizations include the use of multithreading, data parallelization, and SIMD code generation. Generating portable C code requires MATLAB Coder.

Functionality being removed or changed

`im2java2d` function will be removed

Warns

The `im2java2d` function will be removed in a future release. There is no replacement for this function.

R2019b

Version: 11.0

New Features

Bug Fixes

Big Images: Process images that are too large to fit in memory

Read and process images that are too large to fit in memory by using a `bigimage` object. A `bigimage` supports images with one or multiple resolution levels.

Manage a collection of image blocks that belong to one or more `bigimage` objects by using a `bigimageDatastore`.

Display big image data by using the `bigimageshow` function.

Deep Learning Data Preprocessing: Perform additional image augmentations

You can augment images for network training using more image processing operations.

- Apply randomized color jitter to 2-D RGB images by using the `jitterColorHSV` function.
- Calculate randomized affine transformations, including reflection, rotation, rescaling, and translation, by using the `randomAffine2d` and `randomAffine3d` functions. Apply the transformations to images by using the `imwarp` function.
- Crop 2-D and 3-D images from a random position in the image by using the `randomCropWindow2d` and `randomCropWindow3d` function, respectively.
- Crop 2-D and 3-D images from the image center by using the `centerCropWindow2d` and `centerCropWindow3d` function, respectively.

Random Patch Extraction Datastore: Extract patches from 3-D data and transformed datastores, and train in parallel

You can now use a `randomPatchExtractionDatastore` object to extract random patches from 3-D volumetric input. You can also extract patches from `TransformedDatastore` objects that have an underlying datastore of type `ImageDatastore` or `PixelLabelDatastore`.

`randomPatchExtractionDatastore` now supports parallel training (requires Parallel Computing Toolbox).

Deep Learning: Added example using deep neural networks

The Deep Learning Classification of Large Multiresolution Images example shows how to train a neural network to classify very large multiresolution images that do not fit in memory by using `bigimage` and `bigimageDatastore`.

inpaintExemplar Function: Fill damaged regions in images with exemplar-based inpainting

The `inpaintExemplar` function performs exemplar-based inpainting for object removal and region filling in 2-D grayscale and RGB images. You can interactively select a region by using ROI Creation Convenience Functions, such as `drawfreehand`. After selecting a region, perform image inpainting of that region by using the `inpaintExemplar` function. For an example, see Interactive Image Inpainting Using Exemplar Matching.

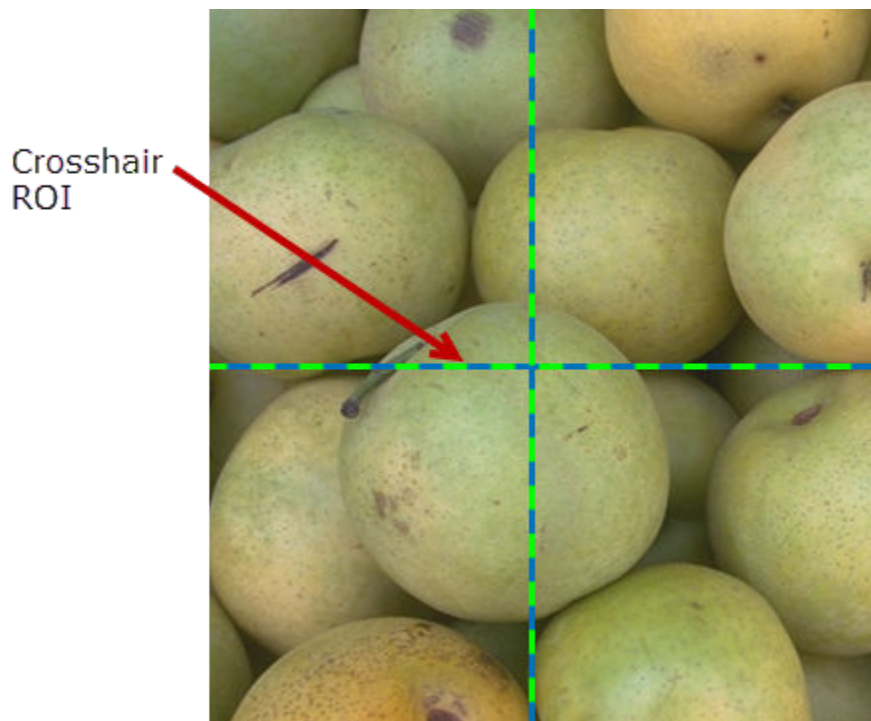
DICOM Volume: Construct isotropic volume from DICOM images

The `dicomreadVolume` function can now construct an isotropic volume from a given set of DICOM images. Use the new name-value pair `'MakeIsotropic'`, `true` in `dicomreadVolume` function to return isotropic volume at the output.

ROI Tools: Create crosshair shape and other enhancements

The ROI tools include these enhancements.

- Crosshair ROI — Use the `drawcrosshair` function or `Crosshair` object to create a crosshair ROI in an axes.

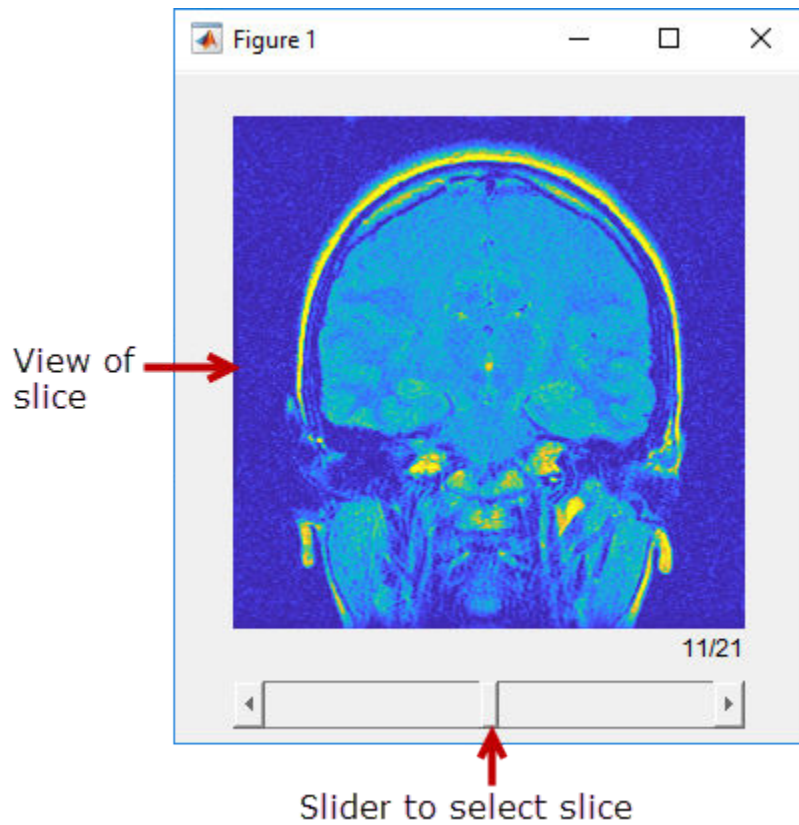


- `wait` function — All ROI objects now support the `wait` method so that the objects can be used in scripts. For example, using the `wait` function, you can enable users of your script to make the initial placement of the ROI, adjust the ROI and accept it, and then use the ROI position in the script to create a mask.
- `reduce` function — The polygon, polyline, freehand, and assisted freehand ROIs now support the `reduce` method. The `reduce` function reduces the number of points used to define the ROI by using the Ramer-Douglas-Peucker algorithm. For an example, see `Subsample` or `Simplify a Freehand ROI`.
- Support for `UIAxes` — You can now create an ROI in a `UIAxes`. For information about limitations, see `Using ROIs in Apps Created with App Designer`.

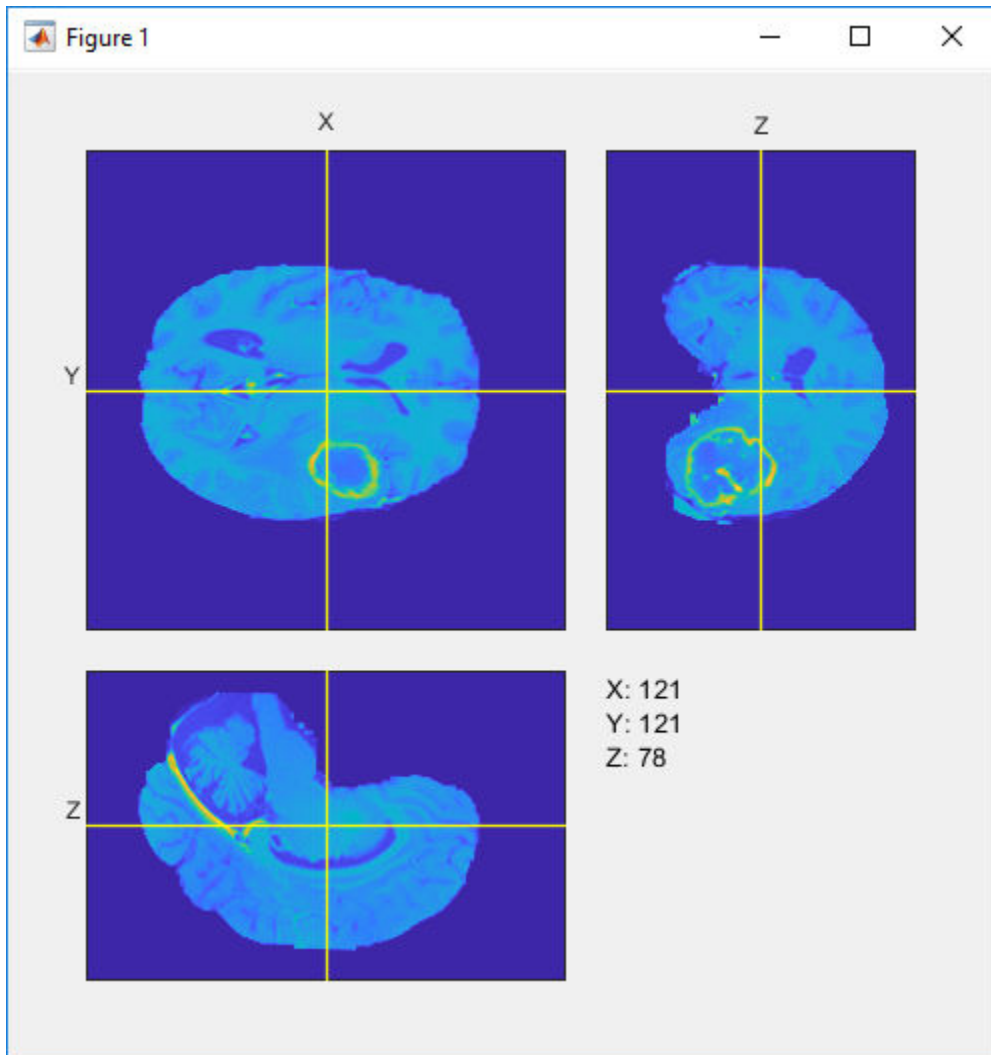
View 3-D Volumes as Slice Planes

Use `sliceViewer` and `orthosliceViewer` to view 3-D grayscale or RGB volumes as slice planes.

`sliceViewer` displays the center slice plane when it opens. You can navigate through the other slices by moving the slider at the bottom of the figure.

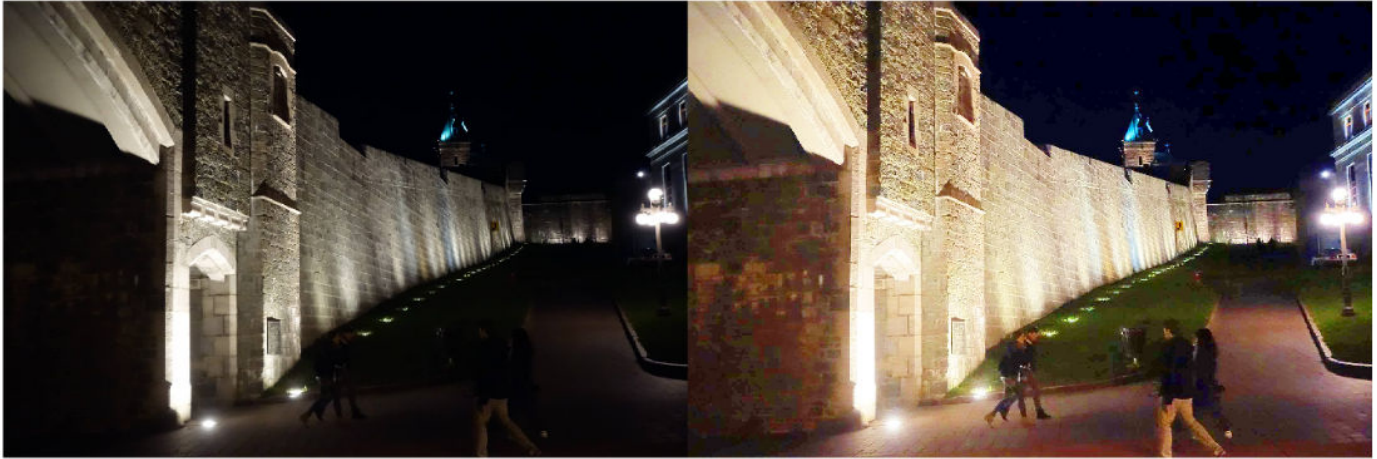


`orthosliceViewer` displays three views of the 3-D volume: the x -, y -, and z -planes. You can navigate through each view of the 3-D volume by moving the crosshair in the figure.



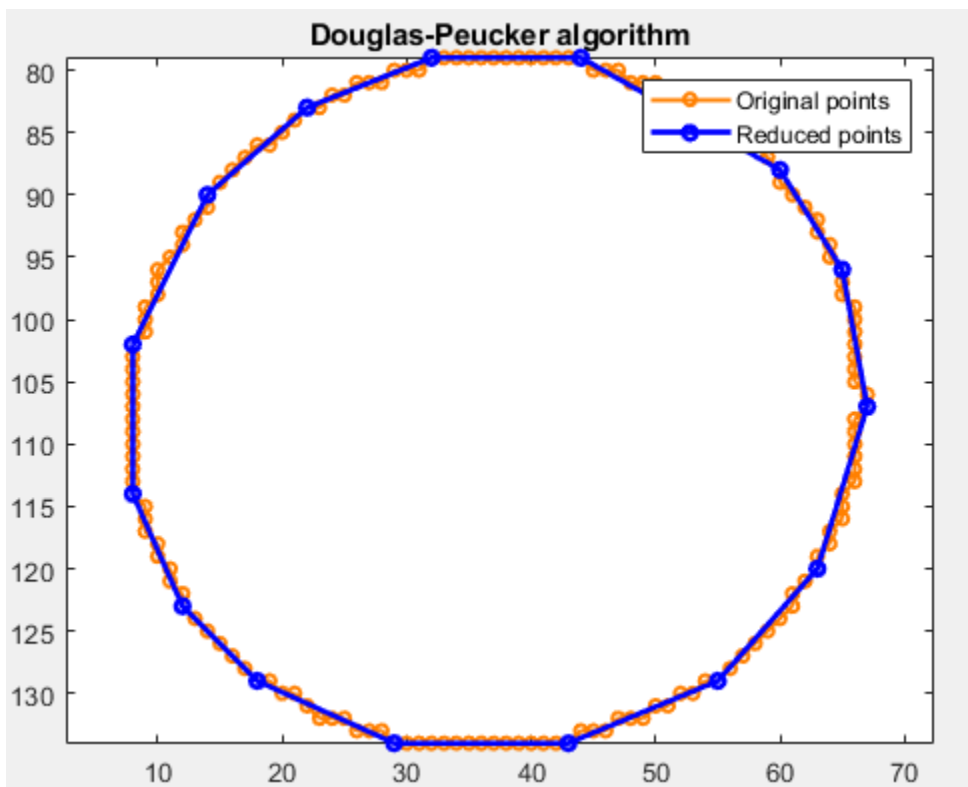
imlocalbrighten Function: Brighten dark areas of images

The `imlocalbrighten` function brightens dark areas of an image.



reducepoly Function: Reduce density of points in ROIs

The `reducepoly` function reduces the density of points in an ROI, by using the Ramer-Douglas-Peucker algorithm.



Volume Viewer App: Create new session, export visualization settings, and other enhancements

The **Volume Viewer** app includes several new capabilities:

-
- **New Session** button — Use this button to clear all the data currently in the app.
 - **Export** button — Use this button to save the current rendering and camera configurations in the window. This option returns the configuration in a structure. To replicate a view in **Volume Viewer**, pass this structure to the `volshow` function.
 - Additional alphasmap presets — In the Rendering Editor, use several new preset alphasmaps for viewing CT and MRI data, including maximum intensity projection alphasmaps.

imshow Function: Specify interpolation method

The `imshow` function now enables you to choose the interpolation method used when scaling an image. You can choose either nearest-neighbor ('nearest') or bilinear ('bilinear') interpolation for the 'Interpolation' Name/Value pair argument.

volshow Function: Control lighting in volume rendering

The `volshow` function now enables you to control lighting in rendering volumes, similar to the lighting control in the **Volume Viewer** app.

affineOutputView Function: Control view of warped images

The `affineOutputView` function creates a spatial referencing object that can be used with the 'OutputView' argument of `imwarp` to control the output limits and grid spacing of a warped image.

Image Cropping: Crop 3-D volumes and crop using spatial referencing

The `imcrop3` function enables you to crop 3-D volumetric images.

The `Rectangle` and `Cuboid` objects store the spatial extents of 2-D and 3-D cropping windows, respectively. The `imcrop` function now accepts a `Rectangle` input argument that specifies the size and position of a 2-D cropping window. Likewise, the `imcrop3` function accepts a `Cuboid` input argument that specifies the size and position of a 3-D cropping window.

Support for Categorical Data

These functions now accept categorical data as input and output categorical data as outputs: `imcrop`, `imresize`, `imresize3`, and `imwarp`.

C Code Generation: Generate code from the imregcorr function using MATLAB Coder

The `imregcorr` function supports the generation of C code using MATLAB Coder. If you choose the generic MATLAB Host Computer target platform, the function generates code that uses a precompiled, platform-specific shared library.

R2019a

Version: 10.4

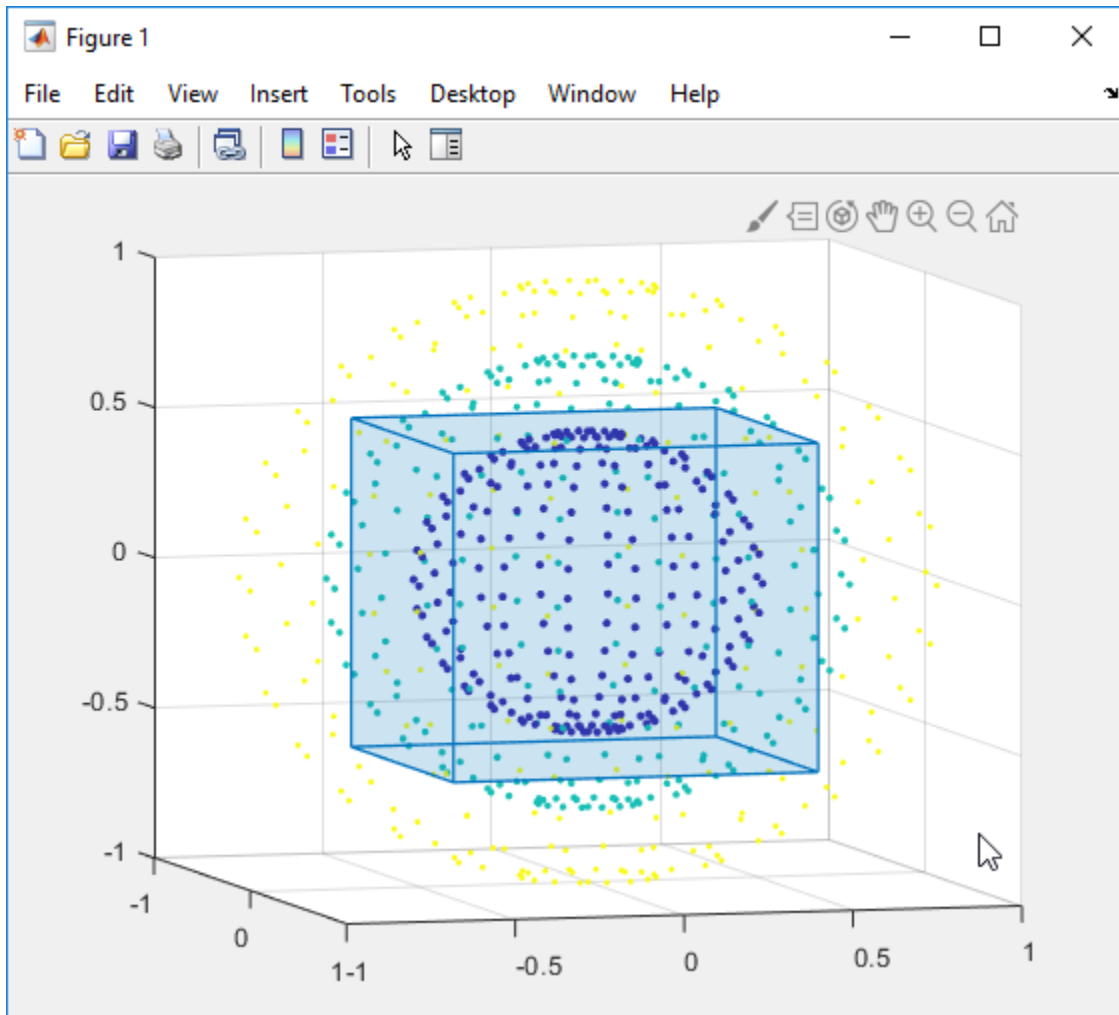
New Features

Bug Fixes

Compatibility Considerations

ROI Creation Functions: New cuboid shape added

You can create cuboidal ROIs for 3-D volumes with the new `drawcuboid` function or the `images.roi.Cuboid` class. As with the other ROIs, cuboidal ROIs support properties, methods, and events that let you customize their appearance and behavior.



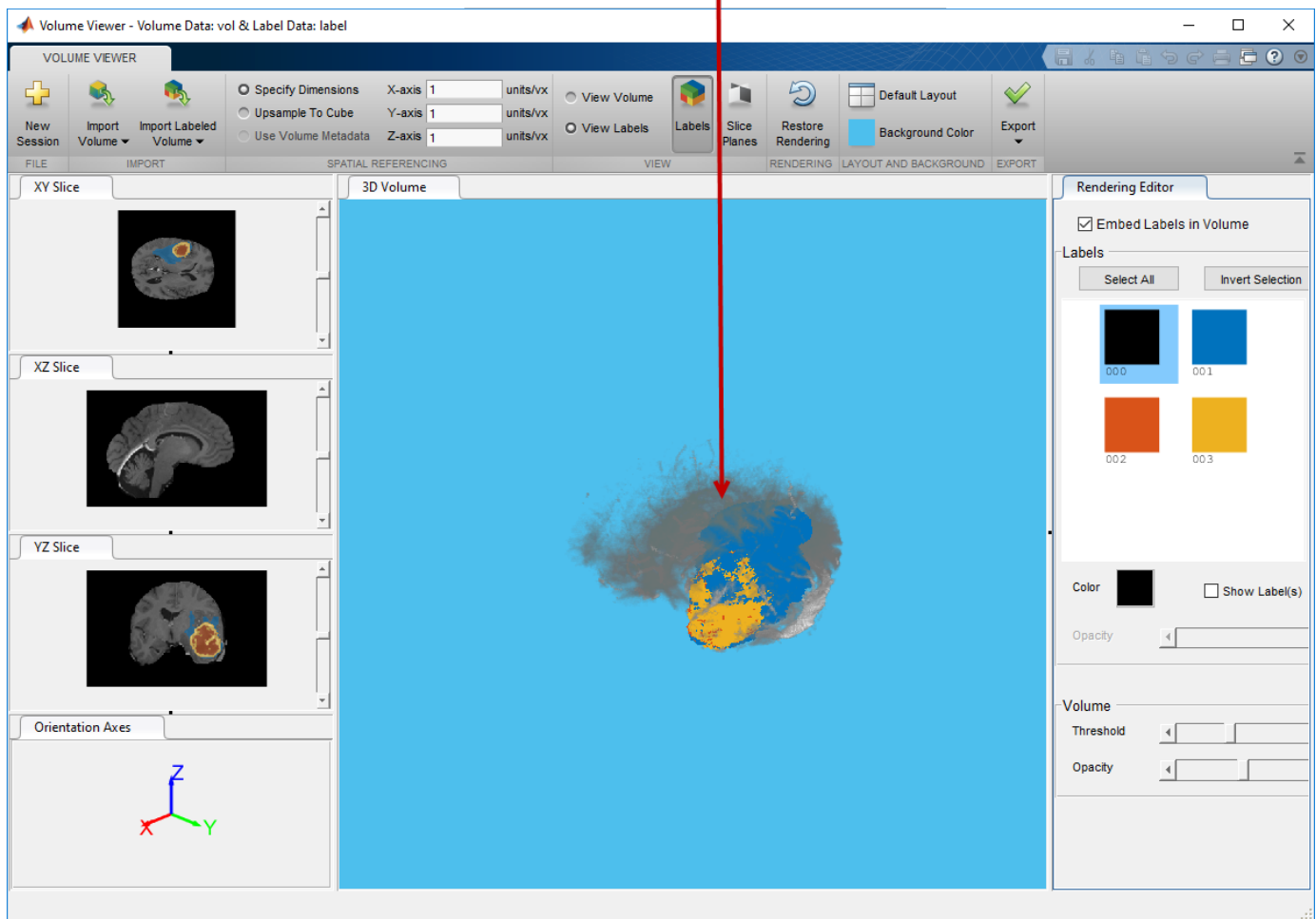
ROI Creation Functions: New `bringToFront` Function

You can now change the visual stacking order of ROIs using the new `bringToFront` function.

Enhanced Volume Display: View labeled volumes and specify colormap

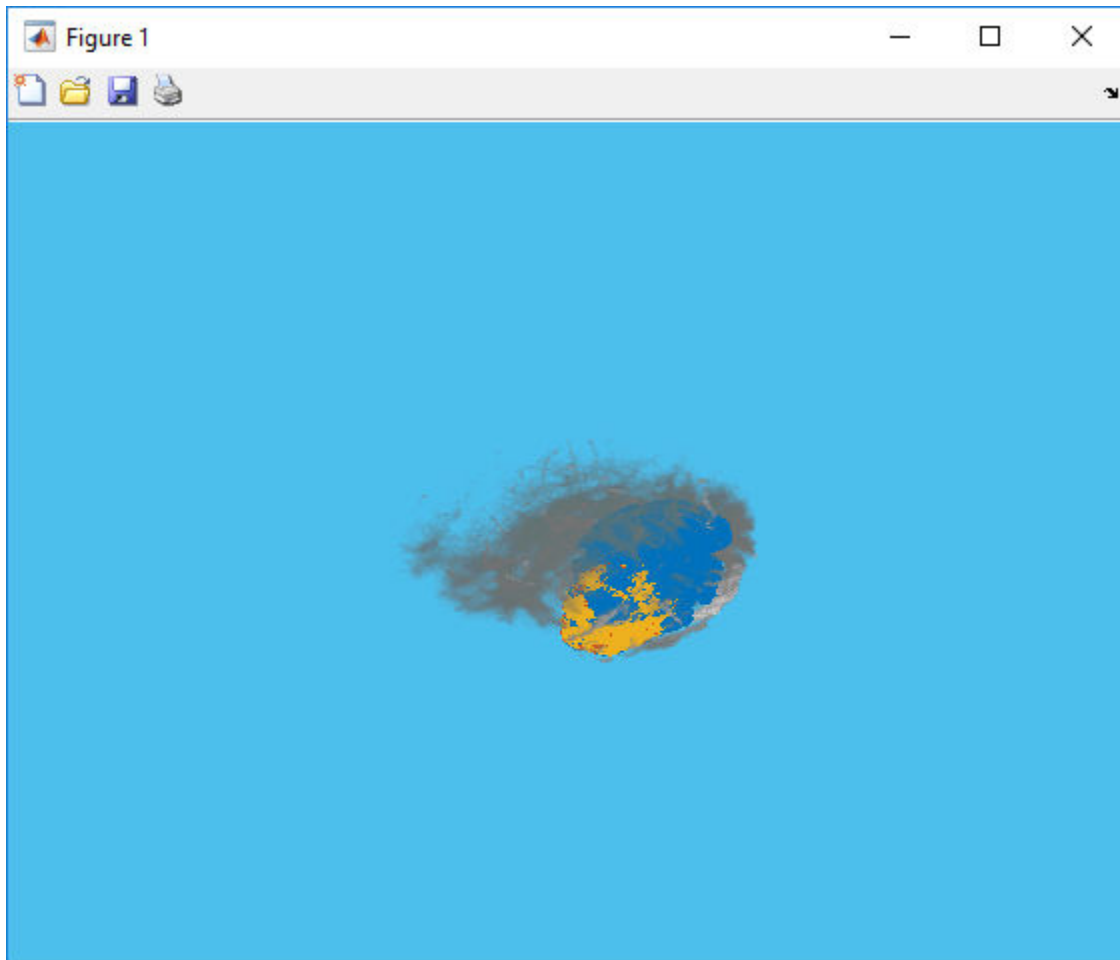
The **Volume Viewer** app now supports the display of 3-D labeled (segmented) volumes. You can also view a labeled volume overlaid on an intensity volume, to view the labels in the context of the original volume.

View labeled volume and intensity volume together.



Volume Viewer now enables you to change the colormap of the displayed intensity volume data. You can select a built-in MATLAB colormap or specify your own colormap.

Instead of using **Volume Viewer**, you can also display 3-D labeled volumes in a figure by using the `labelvolshow` function. As with **Volume Viewer**, you can also view a labeled volume overlaid on an intensity volume.



Deep Learning: Added example using deep neural networks

The 3-D Brain Tumor Segmentation using Deep Learning example shows how to train a neural network to perform semantic segmentation of a 3-D volume.

Measurements of Region Properties: Measure circularity and Feret properties

The `regionprops` function now measures the circularity and Feret properties of regions in a binary image. To measure the circularity, include 'Circularity' when specifying the `properties` input argument. To compute the minimum or maximum Feret properties, include 'MinFeretProperties' or 'MaxFeretProperties', respectively, when specifying the `properties` input argument.

Alternatively, you can measure the minimum or maximum Feret properties of regions in a binary image by using the added `bwferet` function. This table shows the minimum and maximum Feret properties.

Minimum Feret Properties	Maximum Feret Properties
Minimum Feret diameter	Maximum Feret diameter

Angle of minimum Feret diameter	Angle of maximum Feret diameter
Endpoint coordinates of minimum Feret diameter	Endpoint coordinates of maximum Feret diameter

NIfTI File Format Enhancements: Read and write neuroscience image volumes in the NIfTI-2 file format

NIfTI functions `niftiinfo`, `niftiread`, and `niftiwrite` now support NIfTI-2 file formats.

- The metadata returned by the `niftiinfo` function now includes an additional field called `Version`. The value of this field is either `'NIfTI1'` or `'NIfTI2'`, depending on the input file format.
- The `niftiwrite` function now includes an additional name-value pair `Version`. To write a file in NIfTI-2 file format, set `Version` to `'NIfTI2'`.

Camera Response: Estimate real-world illumination as a function of pixel intensity

The `camresponse` function calculates a camera's response function using a series of images captured with different exposure times. The camera response function characterizes the sensitivity of a camera sensor by mapping pixel intensities in an input image to the amount of real-world illumination reaching the sensor.

The `makehdr` function now accepts an optional camera response argument. Providing a camera response argument can improve the accuracy of a high dynamic range image created from a series of low dynamic range images.

inpaintCoherent Function: Fill damaged regions in images with coherence transport based inpainting

The `inpaintCoherent` function performs coherence transport based inpainting for object removal and region filling in 2-D grayscale and RGB images.

burstinterpolant Function: Generate a high-resolution image from a burst of lower resolution images

The `burstinterpolant` function generates a high-resolution image from a set of low-resolution images captured in burst mode. The low-resolution images can be 2-D grayscale or RGB color images.

rgb2lightness Function: Convert an RGB image to a lightness image

The `rgb2lightness` function converts an RGB image to a lightness image. The lightness image corresponds to the lightness component in the CIE 1976 $L^*a^*b^*$ color space.

Performance Improvements: Performance enhancements for image warping

The performance of the following function has been improved:

- `imwarp` (2-D and 3-D)

C Code Generation: Generate code from four additional functions using MATLAB Coder

The following table lists the Image Processing Toolbox functions that have been enabled for code generation in this release. For all target platforms, these functions generate C code. For a complete list of Image Processing Toolbox functions that support code generation, see [Functions Supporting Code Generation](#).

<code>affine3d</code>	<code>inpaintCoherent</code>	<code>rgb2lightness</code>
-----------------------	------------------------------	----------------------------

In addition, the following function that already supported code generation has additional capabilities. This function supports the generation of C code using MATLAB Coder. Note that if you choose the generic `MATLAB Host Computer` target platform, then the function generates code that uses a precompiled, platform-specific shared library.

Function	New Capability
<code>regionprops</code>	Supports the circularity property for code generation.

Functionality being removed or changed

The `imshow` function now displays large images passively

Behavior change

The `imshow` function now handles large images passively. The function does not return a warning message while displaying large images at the largest magnification that fits on the screen. You can modify the initial magnification value by changing the preferred percentage value in the Image Processing Toolbox Preferences dialog box. To access the dialog box, click **Preferences** on the **Home** tab in the MATLAB desktop, or call the `iptprefs` function.

The `imfindcircles` function uses new filter size for logical images

Behavior change

Starting in R2019a, the `imfindcircles` function uses a 5-by-5 filter size for smoothing logical images. `imfindcircles` may now return a different answer than in previous releases, when the filter size was 6-by-6. For example, in some instances, the function may return a different number of circles.

R2018b

Version: 10.3

New Features

Bug Fixes

Compatibility Considerations

Random Patch Extraction Datastore: Extract random image patches to split up large images for deep learning workflows

The `randomPatchExtractionDatastore` object extracts randomly-positioned patches from training images and corresponding patches from images representing the desired network output, for training deep neural networks. The `randomPatchExtractionDatastore` object can also extract randomly-positioned patches from ground truth images and corresponding patches from pixel label data, for training semantic segmentation networks.

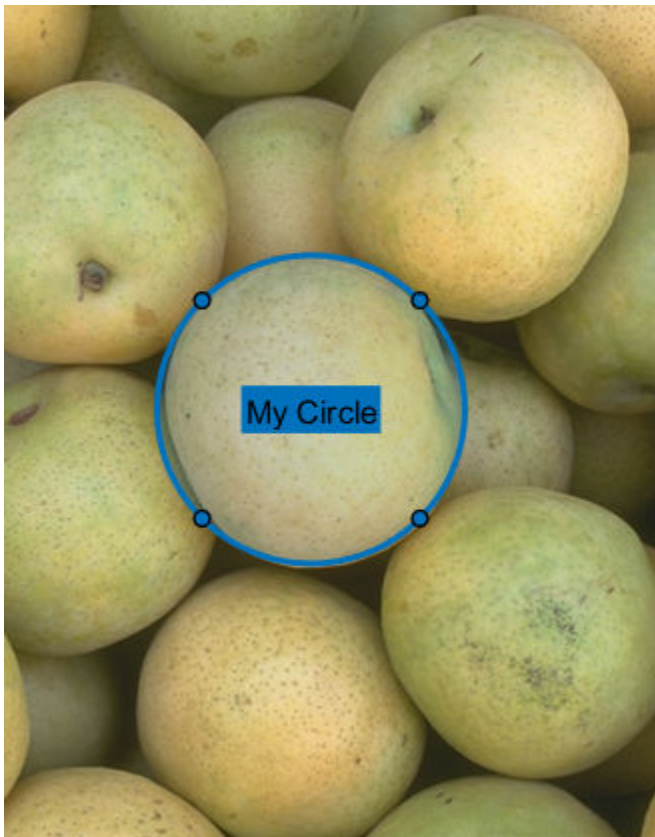
The Single Image Super-Resolution Using Deep Learning example has been updated to use a random patch extraction datastore, instead of a custom mini-batch datastore, as a source of training images.

Deep Learning: Added example using deep neural networks

The Image Processing Operator Approximation Using Deep Learning example shows how to train a neural network to transform an image such that the resulting image resembles the output of a traditional image processing pipeline.

New Set of ROI Creation Functions

The toolbox includes a new set of region-of-interest (ROI) creation functions. Use these functions to create ROIs of many shapes, including circular or polygonal ROIs and ROIs drawn as freehand shapes. A new type of ROI, called assisted freehand, lets you draw a freehand ROI that snaps to edges of existing objects in the image.



The functions create ROI objects. You can change properties of the ROI objects to modify the appearance and behavior of the ROI. For more advanced workflows, such as designing interactive apps, the ROI objects support events and listeners.

ROI Creation Functions	ROI Object	Description of ROI
<code>drawassisted</code>	<code>images.roi.AssistedFreehand</code>	Freehand ROI that snaps to edges of existing objects in the image
<code>drawcircle</code>	<code>images.roi.Circle</code>	Circular ROI
<code>drawellipse</code>	<code>images.roi.Ellipse</code>	Ellipsoid ROI
<code>drawfreehand</code>	<code>images.roi.Freehand</code>	Freehand ROI that follows the path of the mouse
<code>drawline</code>	<code>images.roi.Line</code>	Linear ROI that consists of a single line segment
<code>drawpoint</code>	<code>images.roi.Point</code>	Point ROI
<code>drawpolygon</code>	<code>images.roi.Polygon</code>	Polygonal ROI that consists of a closed set of line segments
<code>drawpolyline</code>	<code>images.roi.Polyline</code>	Polyline ROI that consists of an open set of line segments
<code>drawrectangle</code>	<code>images.roi.Rectangle</code>	Rectangular ROI

You can use the ROI objects as an input to four new functions. The `createMask` function creates a mask with pixels inside the specified ROI set to `true` and pixels outside the ROI set to `false`. The `inROI` function queries if points are within the ROI. The `draw` and `beginDrawingFromPoint` functions begin an interactive drawing mode for the ROI object, maintaining the appearance of the displayed ROI.

These new ROI functions are recommended over the existing set of ROI functions (see “Old ROI classes are not recommended” on page 9-5).

Volume Show: Visualize 3-D image volumes using the `volshow` command

The `volshow` function displays a 3-D grayscale image volume in a figure.

Image Segmentation: Segment 2-D images and N-D volumes using k-means clustering

The `imsegkmeans` and `imsegkmeans3` functions segment images or volumes using k-means clustering.

Geometric Transformation Objects: Represent and apply custom 2-D and 3-D geometric transformations

The `geometricTransform2d` and `geometricTransform3d` objects define 2-D and 3-D geometric transformations using point-wise mapping functions. The objects enable you to apply custom inverse and forward geometric transformations to images.

fspecial3: Create predefined 3-D filters

The `fspecial3` function creates predefined 3-D filters of various types, including ellipsoidal averaging filters, Laplacian of Gaussian filters, and edge-detecting Prewitt and Sobel filters.

imflatfield: Perform flat-field correction

The `imflatfield` function applies flat-field correction to 2-D grayscale and RGB images.

imnlmfilt: Perform non-local means filtering

The `imnlmfilt` function performs edge-preserving, non-local means filtering of 2-D grayscale and RGB images.

imsplit: Split an N-channel image into individual channels

The `imsplit` function creates a set of n images representing each channel in an n -channel image. For an example, see [Display Separated Color Channels of an RGB Image](#).

piqe: Measure image quality using perception-based image quality evaluator (PIQE)

The `piqe` function calculates an opinion-unaware no-reference quality score for natural images using perception-based features.

tonemapfarbman: Reduce dynamic range of HDR images

The `tonemapfarbman` function converts high dynamic range (HDR) images into a format that can be displayed using edge-preserving multi-scale decompositions.

fibermetric: Added 3-D support

The `fibermetric` function now supports 3-D grayscale input volumes.

Processing volumetric input requires a performance improvement, which is achieved by a new default value of the `StructureSensitivity` argument of `fibermetric`. The `maxhessiannorm` function is introduced to help calculate the prior default value of `StructureSensitivity` for 2-D images.

Performance improvements: Performance enhancements for 2-D and 3-D morphology, image warping, and fibermetric

The performance of the following functions has been improved:

- `imerode`
- `imdilate`
- `imopen`
- `imclose`
- `imtophat`
- `imbothat`
- `imdiffusefilt`
- `fibermetric`

C Code Generation: Generate code from three additional functions using MATLAB Coder

The following table lists the Image Processing Toolbox functions that have been enabled for code generation in this release. For all target platforms, these functions generate C code. For a complete list of Image Processing Toolbox functions that support code generation, see List of Supported Functions with Usage Notes.

<code>imsplit</code>	<code>fspecial3</code>
----------------------	------------------------

The function listed in the following table previously only generated C code, and can now also generate C code that uses a platform-specific shared library. To use a shared library, choose the generic MATLAB Host Computer option in the MATLAB Coder configuration settings.

<code>bwdist</code>	
---------------------	--

Functionality being removed or changed

fibermetric calculates new default structural sensitivity

Behavior change

Starting in R2018b, `fibermetric` calculates the default value of the `StructureSensitivity` argument for grayscale image `I` as $0.01 * \text{diff}(\text{getrangefromclass}(I))$. The change in the default value increases computational efficiency and reduces memory usage. These improvements also enable processing 3-D volumetric images.

Previous versions of `fibermetric` defined the default value of `StructureSensitivity` as half of the maximum of the Hessian norm of the image. If you want to reproduce the prior default value for 2-D images, then specify `StructureSensitivity` as $0.5 * \text{maxhessiannorm}(I)$. The `maxhessiannorm` function does not support 3-D input.

Old ROI classes are not recommended

Still runs

Old ROI classes (listed in the first column of the table) are not recommended. Use the new ROI functions and classes instead. The new ROI functions and classes have these advantages.

- More functional capabilities, such as face color transparency.
- Support for events. Use events to respond to changes in your ROI such as moving or being clicked.
- Simplified access to the ROI. The new classes return a single object representing the ROI. In contrast, the old classes return the ROI as a group of lines and patch objects.

There are no plans to remove the old ROI classes.

Functionality	Result	Use Instead	Compatibility Considerations
imellipse	Still runs	drawellipse or drawcircle	If you use the imellipse syntax where you specify the position, replace the position vector with the name-value pairs Center and Semiaxes . To specify the position and size of the circle, use the Center and Radius name-value pairs.
imfreehand	Still runs	drawfreehand or drawassisted	None
imline	Still runs	drawline	If you use the imline syntax where you specify the position, replace the position vector with the Position name-value pair.
impoint	Still runs	drawpoint	If you use the impoint syntax where you specify the position, replace the position vector with the Position name-value pair.
impoly	Still runs	drawpolygon or drawpolyline	If you use the impoly syntax where you specify the position, replace the position vector with the Position name-value pair.

Functionality	Result	Use Instead	Compatibility Considerations
imrect	Still runs	drawrectangle	If you use the imrect syntax where you specify the position, replace the position vector with the Position name-value pair.

R2018a

Version: 10.2

New Features

Bug Fixes

Compatibility Considerations

Deep Learning: Added examples using deep neural networks

The following featured examples show how to solve image processing problems by using deep neural networks.

- The Single Image Super-Resolution Using Deep Learning example shows how to recover a high-resolution image from a low-resolution image by using a Very-Deep Super-Resolution (VDSR) neural network. The example shows how to set up and train a VDSR network. The example implements a custom mini-batch datastore, called a `vdsrImagePatchDatastore`, that creates batches of upsampled low-resolution patches and corresponding residual patches, with support for shuffling during training.
- The JPEG Image Deblocking Using Deep Learning example shows how to remove JPEG compression artifacts from images by using a DnCNN network. The example shows how to set up and train the DnCNN network. The example implements a custom mini-batch datastore, called a `JPEGimagePatchDatastore`, that extracts patches from input distorted images and computes the target residual images.
- The Semantic Segmentation of Multispectral Images Using Deep Learning example shows how to use perform semantic segmentation of an image with data in seven channels: three infrared channels, three RGB color channels, and a mask. The example shows how to set up and train a U-Net network to perform the segmentation. The example implements a custom mini-batch datastore, called a `PixelLabelImagePatchDatastore`, that extracts patches from the multispectral images and the corresponding labels.

Deep Learning Data Preprocessing: Efficiently read and add noise to images for training and prediction

A `denoisingImageDatastore` preprocesses training images by adding Gaussian noise. You can use a `denoisingImageDatastore` for both training and prediction.

Image Segmenter: Segment images interactively using new techniques including local graph cut

The **Image Segmenter** app now includes the local graph cut segmentation technique. For an example of segmenting an image by using the Image Segmenter, see Image Segmentation Using the Image Segmenter App.

Edge-Aware Filtering: Smooth images and reduce noise while preserving edge sharpness with bilateral filtering and anisotropic diffusion filtering

The `imdifffilt` function performs anisotropic diffusion filtering of images. The `imbilatfilt` function performs edge-aware bilateral filtering of images by using Gaussian kernels.

blendexposure: Perform exposure fusion

The `blendexposure` function blends images that have different exposures into a single well-exposed image.

imregmtb: Register images using median threshold bitmaps

The `imregmtb` function registers images using the median threshold bitmap technique. This technique is useful for registering images that have camera or scene motion, or that have different exposures.

montage: Display multiple images from ImageDatastore object, and specify size of the image thumbnails displayed

The `montage` function now can display images in an `ImageDatastore`. You can now specify the size of the image thumbnails displayed in the montage. Additionally, `montage` no longer requires *m-by-n-by-1-by-p* inputs for volume inputs. Instead, you can specify an *m-by-n-by-p* array.

Starting in R2018a, there are several changes to the behavior of `montage`. See [Compatibility Considerations](#).

Image Morphology: Perform morphological operations on 3-D volumes, and perform skeletonization on all objects in 2-D image or 3-D binary volume

You can perform morphological operations on 3-D volumes, and perform skeletonization on all objects in a 2-D image or 3-D binary volume.

The `bwmorph3` function performs morphological operations on 3-D volumes.

The `bwskel` function reduces all objects in the 2-D binary image or 3-D binary volume to curved lines, without changing the essential structure of the image.

Performance: Improved performance in functions including 3-D imwarp, 3-D imfilter, entropyfilt, ordfilt2, medfilt2, and bwmorph

The performance of the following functions has been improved:

- `bwmorph`
- `entropyfilt`
- `imfilter` (3-D)
- `imwarp` (3-D)
- `medfilt2`
- `ordfilt2`

C Code Generation: Generate code from one additional function using MATLAB Coder

The following table lists the Image Processing Toolbox function that has been enabled for code generation in this release. For all target platforms, this function generates C code. This function can also generate C code that uses a precompiled, platform-specific shared library (`.dll`, `.so`, or `.dylib`). Using a shared library preserves performance optimizations in this function but limits the target to only those platforms that support MATLAB (see system requirements). For a complete

list of Image Processing Toolbox functions that support code generation, see List of Supported Functions with Usage Notes.

<code>imblatfilt</code> ¹	
--------------------------------------	--

¹ If you choose the generic MATLAB Host Computer option in the MATLAB Coder configuration settings, this function generates C code that uses a precompiled, platform-specific shared library.

Functionality being removed or changed

montage function has several behavior changes

Behavior change

Starting in R2018a, there are several changes to the behavior of `montage`.

Previous Behavior	New Behavior	Compatibility Mode
If you pass a single truecolor (RGB) image to <code>montage</code> , then the function displays a single truecolor image.	<code>montage</code> displays each color channel of an RGB image as three separate grayscale images.	Use the <code>imshow</code> function to display a truecolor image <code>imshow(RGB)</code> .
If one or more of the image files contained an indexed image, then <code>montage</code> used the colormap from the first indexed image file.	<code>montage</code> converts any indexed image into its corresponding RGB version using the internal colormap present in the file.	Explicitly read the colormap of the first indexed image and then use the syntax <code>montage(filename, cmap)</code> .
Each constituent thumbnail preserved the full image size.	Each thumbnail is resized to the specified (or default) <code>ThumbnailSize</code> with appropriate zero-padding.	Use the syntax <code>montage(..., 'Thumbnail', [])</code> .
If you specify a set of indexed images, then <code>montage</code> used the colormap of the first image to set the colormap of the axes. This enabled you to change the colormap of all the images displayed after creating the montage by changing the colormap of the axes.	If you specify a set of indexed images, then <code>montage</code> converts each one to an RGB image using its internal colormap or the colormap specified in the command line. Changing the colormap of the axes has no effect.	Explicitly specify the colormap at time of creation, using the syntax <code>montage(filename, cmap)</code> .

denoisingImageSource object is removed

In 2017b, you could create a `denoisingImageSource` object for training deep learning networks. Starting in R2018a, the `denoisingImageSource` object has been removed. Use a `denoisingImageDatastore` object instead.

A `denoisingImageDatastore` has additional properties and methods to assist with data preprocessing. Unlike `denoisingImageSource`, which could be used for training only, you can use a `denoisingImageDatastore` for both training and prediction.

To create a `denoisingImageDatastore` object, you can use either the `denoisingImageDatastore` function (recommended) or the `denoisingImageSource` function.

denoisingImageSource function will be removed

Still runs

The `denoisingImageSource` function will be removed in a future release. Create a `denoisingImageDatastore` using the `denoisingImageDatastore` function instead.

To update your code, change instances of the function name `denoisingImageSource` to `denoisingImageDatastore`. You do not need to change the input arguments.

R2017b

Version: 10.1

New Features

Bug Fixes

Deep Learning: Denoise images using deep learning techniques

The toolbox includes the new `denoiseImage` function, which estimates the denoised version of a noisy image using a denoising deep neural network. You can use a pretrained network using the new `denoisingNetwork` function, or you can train your own network starting with layers provided by the new `dnCNNLayers` function. These functions require the Neural Network Toolbox™.

3-D Image Processing: Process 3-D volumetric image data with support for several additional functions

The toolbox includes several new functions that enable working with 3-D volumetric data: `regionprops3`, `edge3`, `bwselect3`, `jaccard`, `dice`, `bfscore`, and `dicomreadVolume`. The toolbox includes one new function that enables working with N-D data: `imadjustn`.

Five existing functions now support 3-D volumetric data: `imbinarize`, `adaptthresh`, `niftiinfo`, `niftiread`, and `niftiwrite`.

Image Enhancement: Adjust colors with automatic white balancing, and reduce haze in images

The toolbox includes several new functions that enhance the appearance of images.

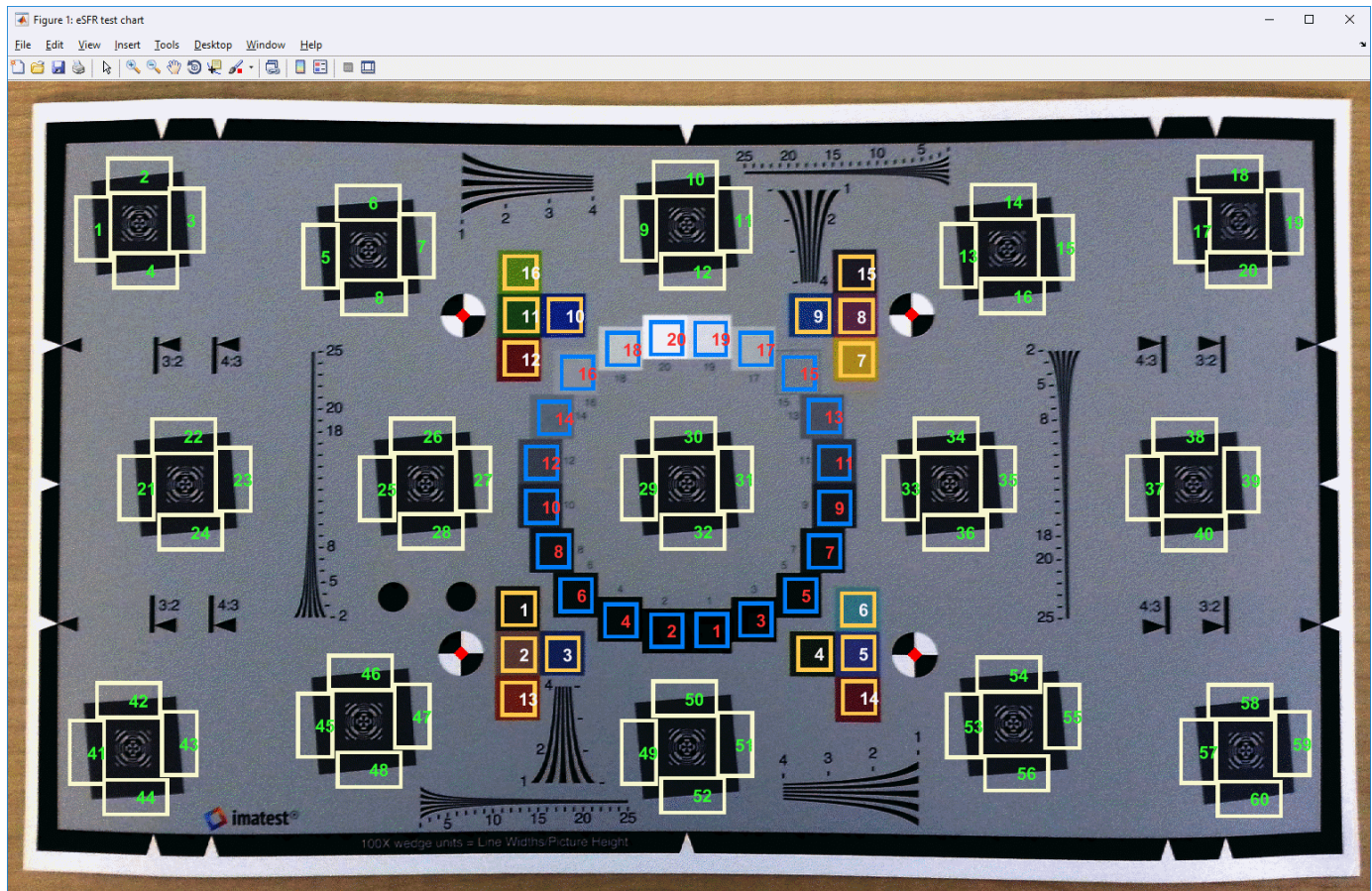
- The new `chromadapt` function adjusts the color balance of an sRGB image according to the scene illuminant. Three new functions, `illumgray`, `illumzca`, and `illumwhite`, estimate scene illumination using the Gray World algorithm, principal component analysis, and the White Patch Retinex algorithm.
- The new `imreducehaze` function reduces atmospheric haze in an RGB or grayscale image.
- The new `lin2rgb` function applies gamma correction to convert linear RGB images to sRGB or Adobe RGB (1998). The new `rgb2lin` function undoes gamma correction to linearize RGB images.

Image Quality Metrics: Measure image quality without a reference image, and model image quality using an eSFR test chart

The toolbox includes several new functions and objects that enable objective measures of quality from images.

- The new `brisque` and `niqe` functions calculate a no-reference quality score for images using trained models with natural scene statistics. You can train custom models using the `fitbrisque` and `fitniqe` functions. The models are stored in `brisqueModel` and `niqeModel` objects.
- The new `eSFRChart` object automatically identifies the slanted edge, gray patch, and color patch regions of interest (ROIs) of an edge spatial frequency response (eSFR) test chart from `Imatest`. You can display the ROIs overlaid on the chart using the `displayChart` function.

You can measure several aspects of the eSFR test chart using the `measureSharpness`, `measureChromaticAberration`, `measureColor`, `measureNoise`, and `measureIlluminant` functions. You can display the measured sharpness using the `plotSFR` function. You can visually compare the measured and expected colors using the `displayColorPatch` and `plotChromaticity` functions.



NIfTI File Format: Read and write neuroscience image volumes in the NIfTI file format

The toolbox includes three new functions that enable reading, writing, and reading image volumes from files in the file format defined by the Neuroimaging Informatics Technology Initiative (NIfTI). The functions are: `niftiinfo`, `niftiread`, and `niftiwrite`. This format is commonly used in the neuroimaging community for CT, MR, and PET data.

Image Segmenter: Segment images using new techniques including texture segmentation

The **Image Segmenter** app now includes several new segmentation techniques including texture segmentation using Gabor filtering. If you also have the Statistics and Machine Learning Toolbox™, the Image Segmenter includes an automatic clustering capability based on K-means processing. In addition, you can now load a pre-existing mask image into the Image Segmenter to continue refining a mask.

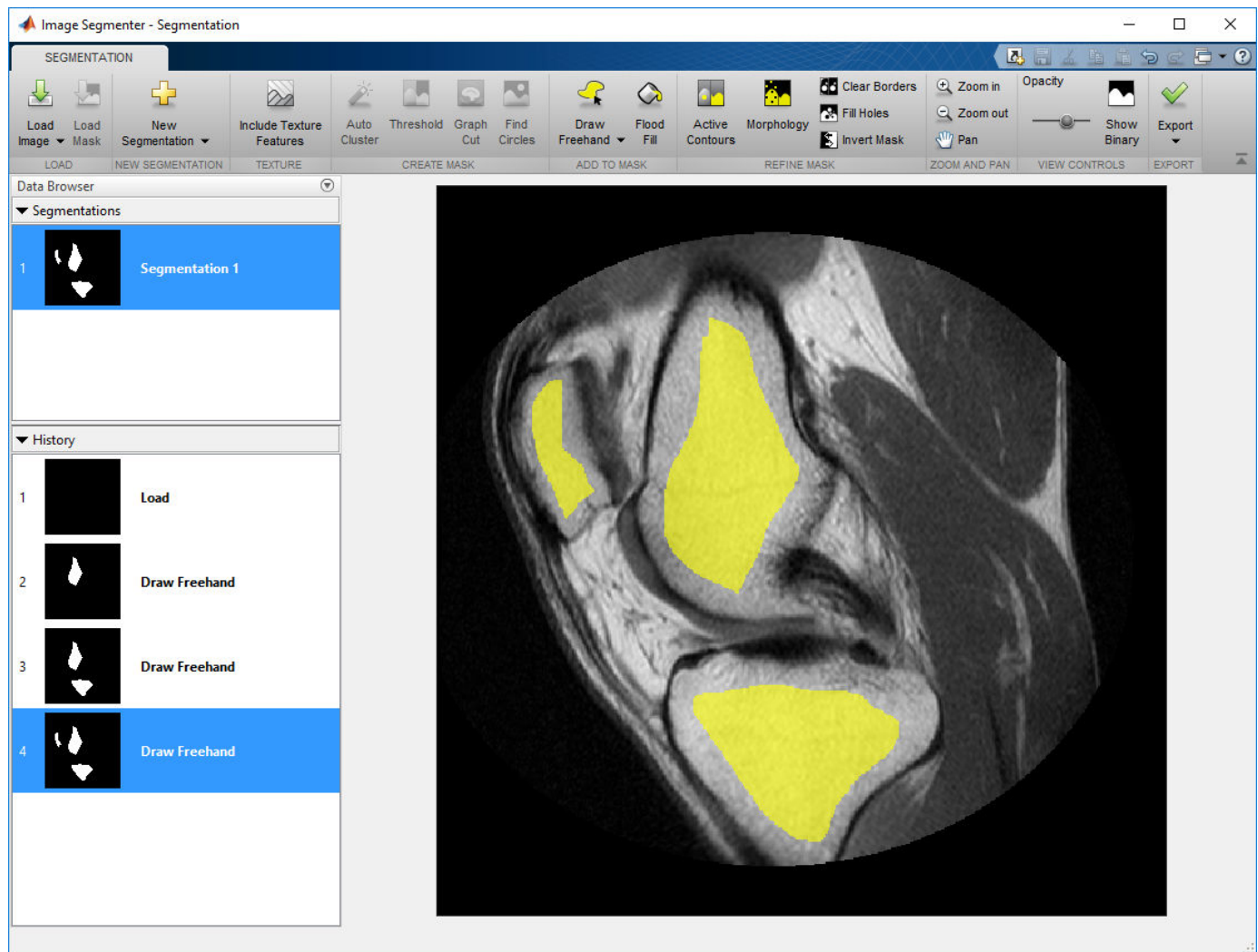


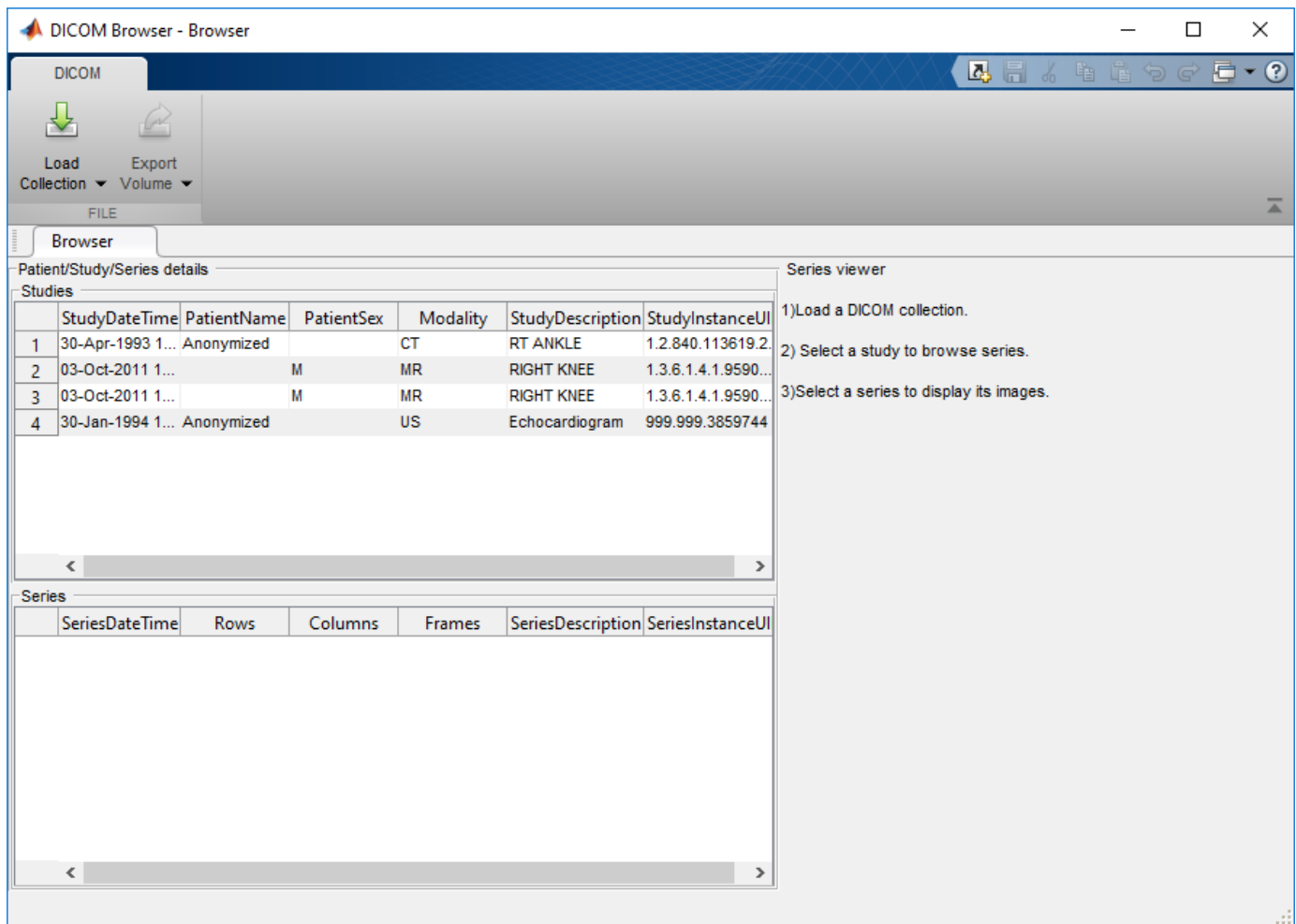
Image Segmentation: Calculate similarity coefficients

The jaccard, dice, and bfscore functions compute similarity coefficients that can be used to evaluate segmentation accuracy.

DICOM Browsing: Explore the contents of DICOM media in a browser and programmatically

The toolbox now includes several new functions and an app that you can use to explore the contents of DICOM files and folders.

The DICOM Browser app (**DICOM Browser**) lets you explore the contents of DICOM media.



In addition, you can read volumetric data from DICOM media using the new `dicomreadVolume` and `dicomCollection` functions. You can also decode DICOM UIDs and parse the DICOMDIR file using `images.dicom.decodeUID` and `images.dicom.parseDICOMDIR` functions.

Image Warper: Transform group of images quickly using the `images.geotrans.Warper` object

The toolbox includes a new object, called Warper, that enables you to apply a geometric transformation to a group of images, all the same size. You identify the images, define the geometric transformation, and then create a Warper object. To apply the transformation to the images, use the `warp` function.

R2017a

Version: 10.0

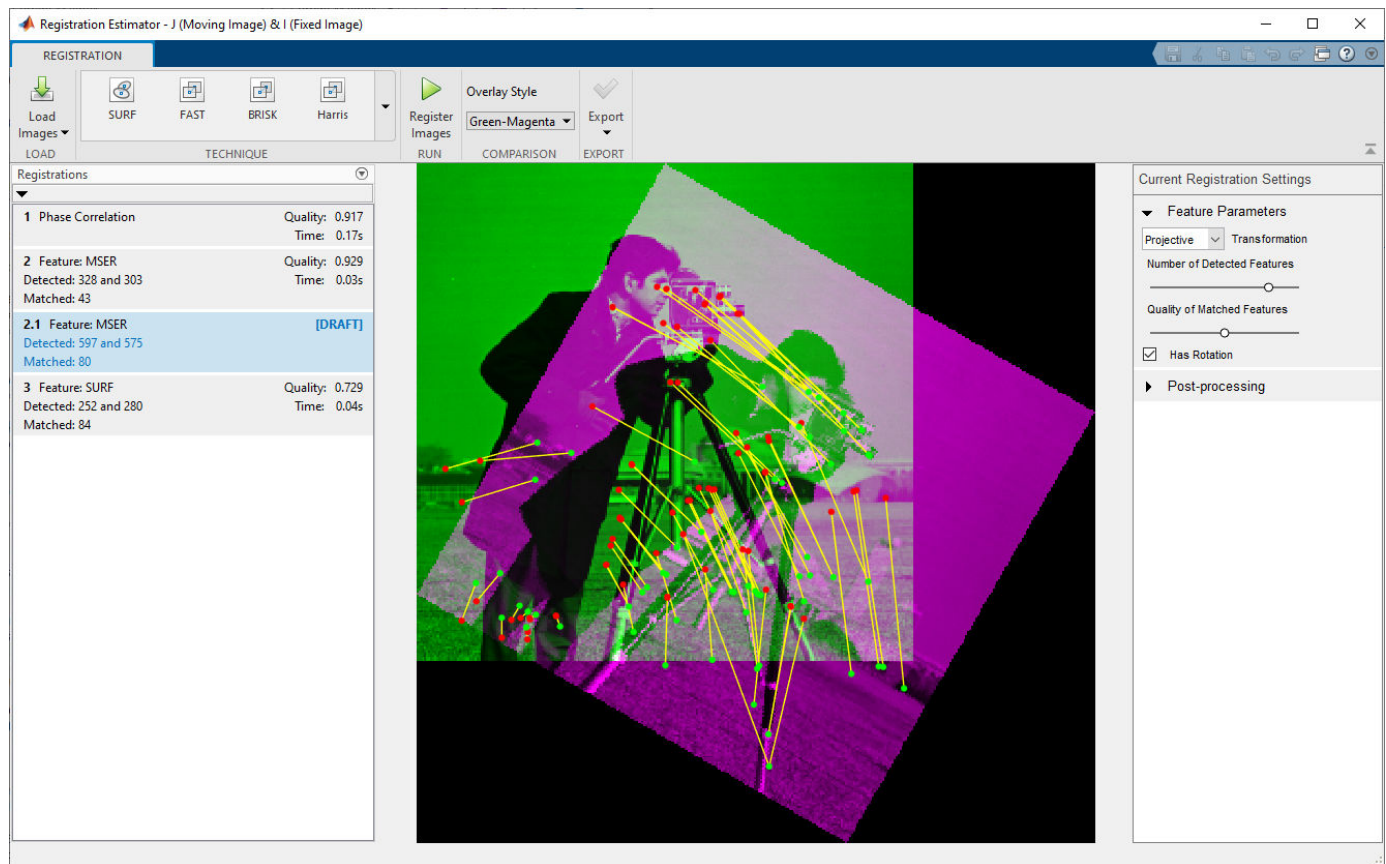
New Features

Bug Fixes

Compatibility Considerations

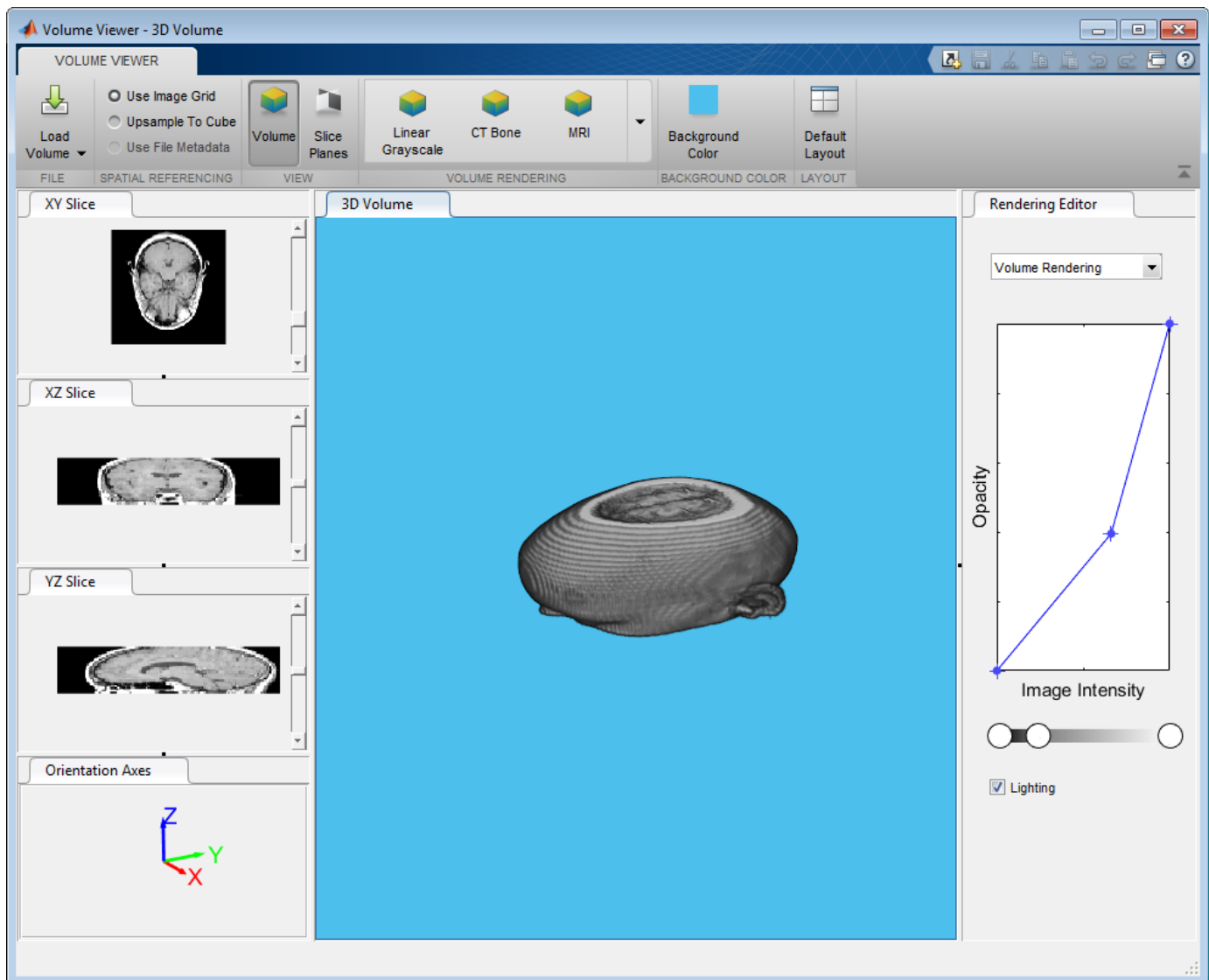
Image Registration App: Explore various registration techniques interactively to align images

The new image registration app, called Registration Estimator, lets you use several different registration techniques to bring two images into alignment. You can use feature-based, intensity-based, or nonrigid approaches. For more information, see Register Images Using the Registration Estimator App.



Volume Viewer App: View and slice 3-D volumetric data

The Volume Viewer app lets you view 3-D volumetric data as a volume or as a set of slices. The Volume Viewer provides a rendering editor that enables you to manipulate the mapping of intensity values to opacity to achieve different views of your 3-D data. The Volume Viewer includes several preset mappings to achieve certain well-known effects, such as emphasizing the visibility of bony structures or revealing inner structures in your data. For more information, see Explore 3-D Volumetric Data with the Volume Viewer App.



3-D Volumetric Data: Process 3-D volumetric image data with support for over a dozen functions

The toolbox now provides more support for processing 3-D volumetric data. This includes several new functions, `imresize3` and `imrotate3`. In addition, existing functions, such as `activecontour`, highlight their support of volumetric data, many with new 3-D examples. To see an example of segmenting a volume, see `Segment Lungs from 3-D Chest Scan` and `Calculate Lung Volume`.

fibermetric: Enhance tubular or elongated structures in images

The toolbox includes the new function, `fibermetric`, that you can use to enhance elongated or tubular structures in an image.

Image Segmenter App: Added support for RGB images and graph cut segmentation

The Image Segmenter app now includes support for RGB images and segmentation using the graph cut technique. Graph cut is a semi-automatic technique of segmenting the foreground from the background in an image. For more information, see [Segment Image Using Graph Cut](#).

lazysnapping: Segmentation technique

The toolbox includes the new function, `lazysnapping`, that you can use to segment an image foreground from the background.

N-D Histograms: Enhance contrast and adjust histograms of N-D images

The toolbox includes a new function, `imhistmatchn`, that allows you to match an N-D histogram to a reference histogram. The toolbox adds N-D support to two other functions, `imhist` and `histeq`, that you can use to calculate the histogram and enhance the contrast of N-D images.

dicomread Supports JPEG Variant

The `dicomread` function now supports JPEG compression, process 2 and 4.

imfilter can return different results for codegen with certain inputs

When using `imfilter`, if you specify a large kernel, a kernel that contains large values, or specify an image containing large values, you can see different results between MATLAB and generated code using `codegen` for floating point data types. This happens because of accumulation errors due to different algorithm implementations.

Functions Being Removed or Changed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
<code>imclose</code> and functions that perform morphological closing using <code>imclose</code> , such as <code>imbothat</code> .	Still works	Not applicable	<code>imclose</code> now pads the input image border by half the size of the structuring element. Padding the image removes border artifacts when there are foreground pixels near the boundary of the input image.

R2016b

Version: 9.5

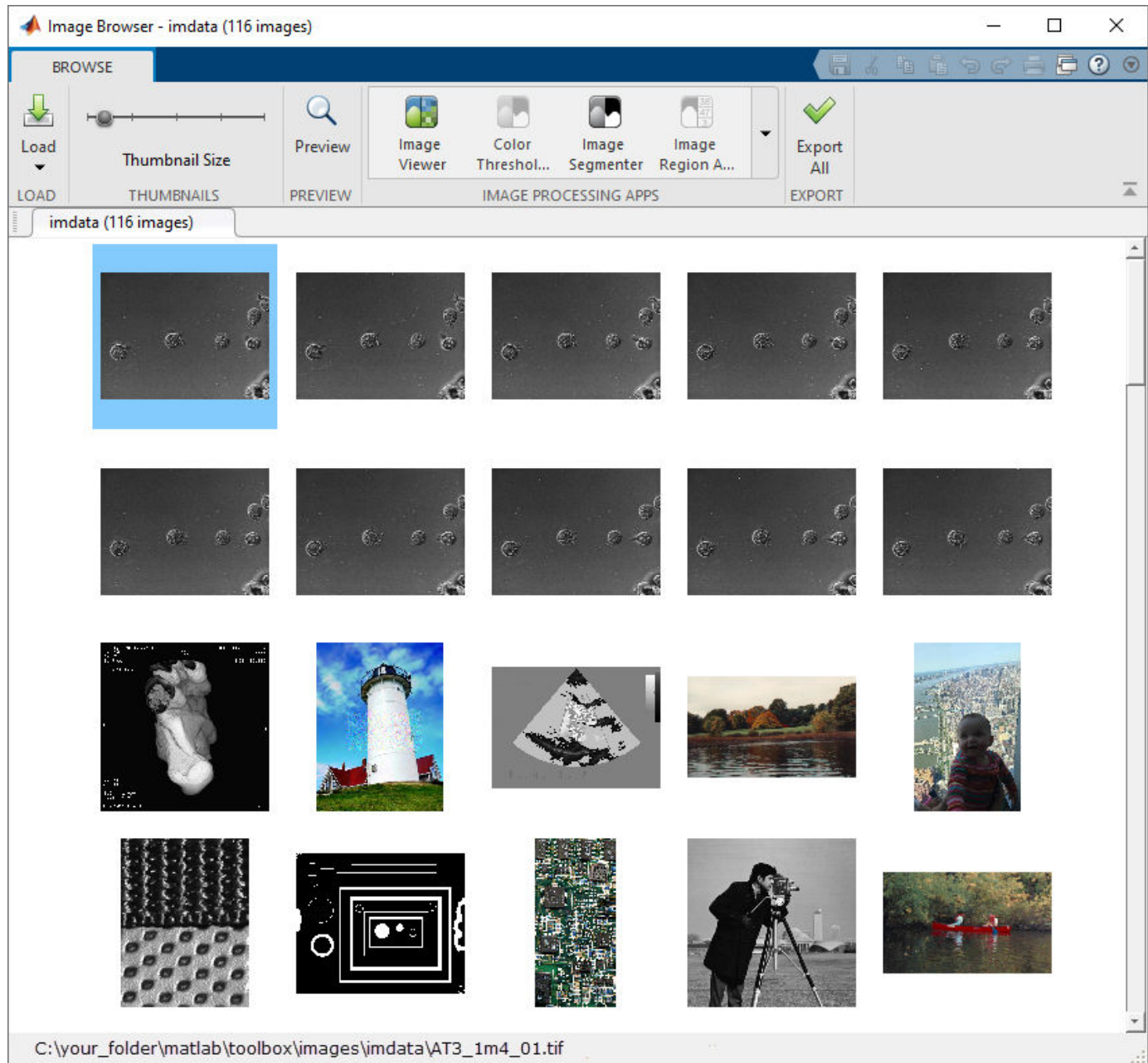
New Features

Bug Fixes

Compatibility Considerations

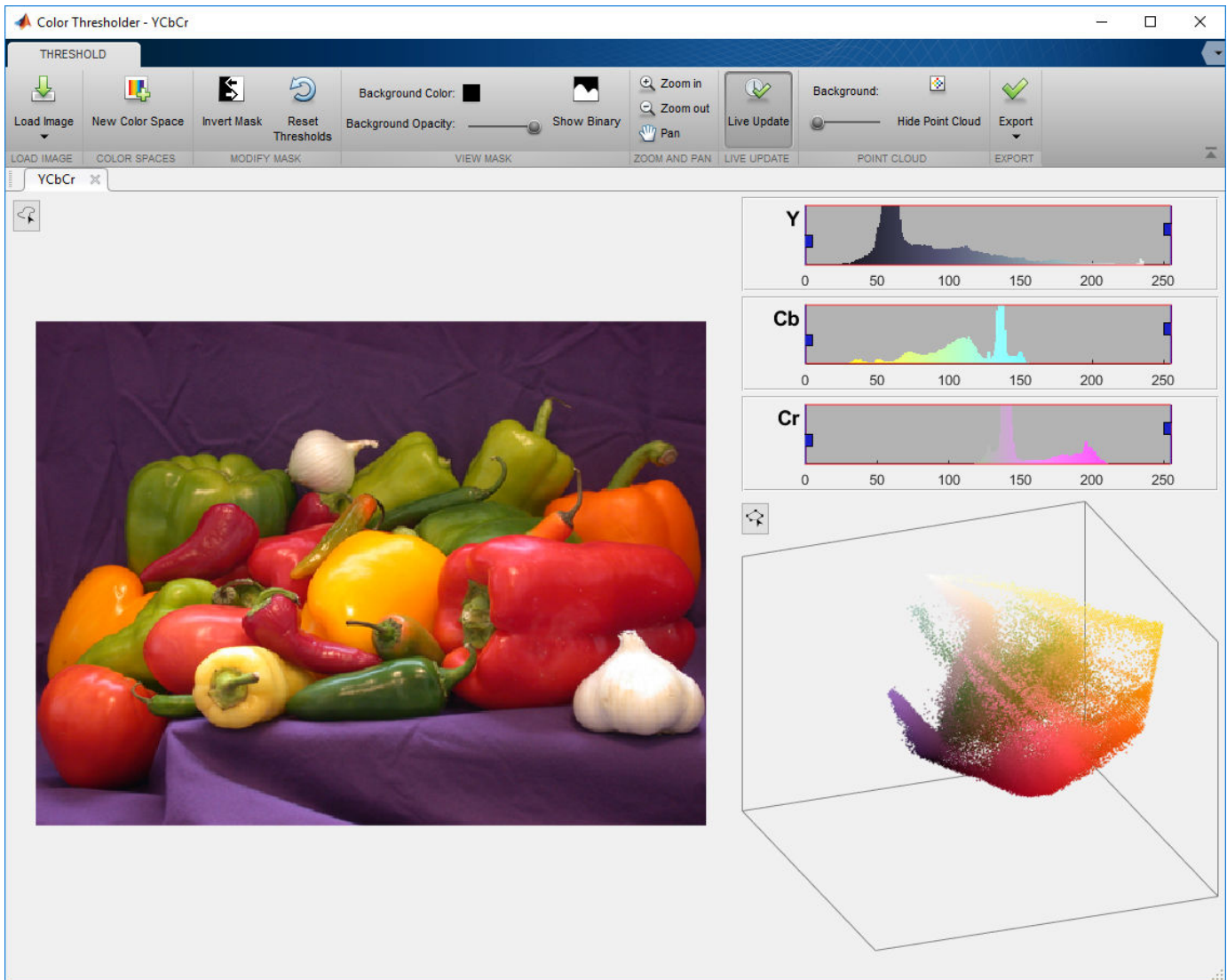
Image Browser App: View multiple images and import selected image into apps

The new Image Browser app lets you view all the images in a folder or an image datastore and lets you import a selected image into an app. For an example, see View Thumbnails of Images in Folder or Datastore.



Color Thresholder App: View color data as point cloud for segmentation

The Color Thresholder app now lets you view image color data as a point cloud in four different color spaces: RGB, HSV, YCbCr, L*a*b*. You can use these point clouds to see which color space provides the clearest separation of color that makes it easy to select particular colors for segmentation. You use the point clouds in conjunction with the existing controls offered for each color space. For an example, see Image Segmentation Using Point Clouds in the Color Thresholder App.



Edge-Aware Filtering: Perform edge-aware filtering based on Laplacian pyramids

The toolbox includes several new functions that provide edge-aware filtering based on Laplacian pyramids. These functions, `locallapfilt`, `localcontrast`, and `localtonemap`, produce high-quality results without introducing halos.



Before and After Images of Local Laplacian Filtering

3-D Superpixels: Use simple linear iterative clustering (SLIC) with volumetric images

The toolbox includes a new function that enables you to perform a simple linear iterative clustering (SLIC) 3-D superpixel segmentation of 3-D volumetric images. Use the `superpixels3` function to obtain the segmentation of the image, using the SLIC superpixel algorithm.

3-D Median Filtering: Apply median filter to volumetric image data

The toolbox includes a new function for 3-D median filtering: `medfilt3`.

colorcloud: Display color information as a point cloud

The toolbox includes the new function, `colorcloud`, that you can use to display color information in an image as a point cloud.

edge: Perform faster Canny edge detection

The `edge` function supports a new parameter, `approxcanny`, that lets you perform Canny edge detection with better performance than the `canny` option, although the `canny` option might provide more precise filtering.

imshow now sets colormap of axes

The `imshow` function now sets the `colormap` property of individual axes, rather than setting the `colormap` of the figure object. This enables graphics objects with different colormaps to be displayed in the same figure. When displaying multiple images in one figure window using `subplot`, each image maintains its own colormap. For an example, see [Display Multiple Images in the Same Figure](#).

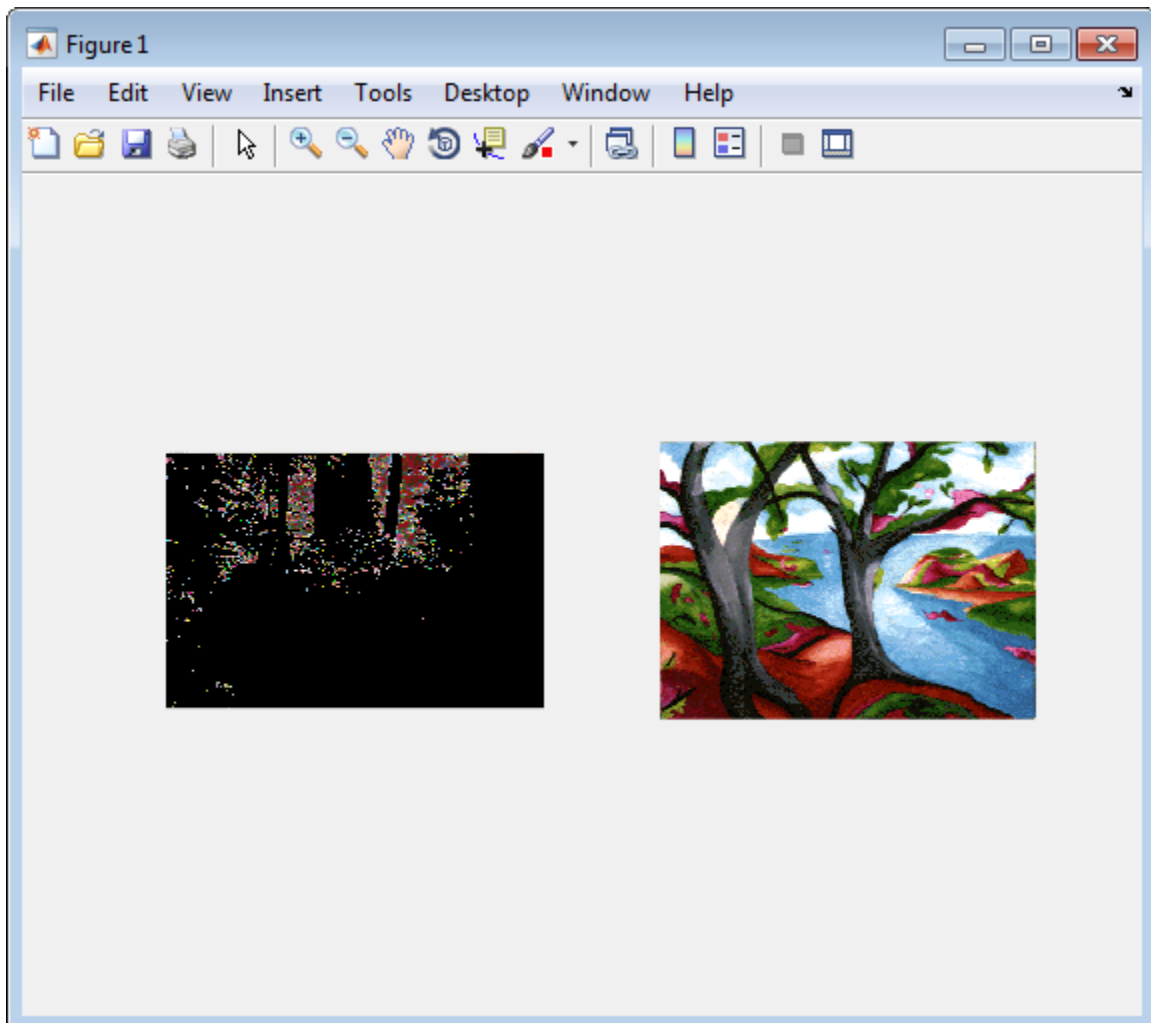
Compatibility Considerations

Certain legacy code may be impacted by the new behavior of `imshow`, including:

- Code that depends on all axes in the figure window having the same colormap as the most recently displayed graphics object.
- Code that gets the colormap of the figure window after using `imshow`.
- Code that sets the colormap of the figure window after using `imshow`.

You can reproduce the prior behavior of `imshow` in subplots by providing each axes in the figure window with the same colormap, determined by the most recent graphics object displayed. For example, you can use this syntax to display two images.

```
[X1,map1]=imread('forest.tif');  
[X2,map2]=imread('trees.tif');  
subplot(1,2,1), imshow(X1,map1)  
subplot(1,2,2), imshow(X2,map2)  
cm = colormap(gca);  
subplot(1,2,1), colormap(gca,cm)  
subplot(1,2,2), colormap(gca,cm)
```



Note how the first image displayed, X1, appears dark after applying the colormap of the second image.

nitfread Supports JPEG2000 Compression

The `nitfread` function now supports NITF files that use JPEG2000 compression.

imregdemons Supports 3-D Images on GPU

The `imregdemons` function now supports 3-D images when run on a GPU.

Contrast Adjustment Tool Now Supports Rsets

The Contrast Adjustment tool that is part of the Image Viewer app now supports reduced resolution files (Rsets).

Specify Initial Folder Opened in Open Image Dialog Box

The `imgetfile` function now supports the new 'InitialPath' option that you can use to specify the folder displayed when the Open Image dialog box opens.

Functions Being Removed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
subimage	Still works	Use <code>imshow</code> instead.	Replace instances of <code>subimage</code> with the <code>imshow</code> function.

R2016a

Version: 9.4

New Features

Bug Fixes

Compatibility Considerations

superpixels: Use simple linear iterative clustering (SLIC) to group pixels for efficient segmentation of color and grayscale images

The toolbox includes several new functions that enable you to perform a simple linear iterative clustering (SLIC) superpixel segmentation of grayscale and color images, and then visualize the segmentation. Use the `superpixels` function to obtain the segmentation of the image, using the SLIC superpixel algorithm. Then, visualize the segmentation using the `boundarymask` function. You can view the segmentation boundaries overlaid on the original image using the `imoverlay` function. You can also obtain lists of the pixels in each segmented region using the `label2idx` function.

Image Segmenter: Segment images using new techniques, including flood-fill and adaptive thresholding

The Image Segmenter app now includes more tools to refine your segmentation, including flood-fill and adaptive thresholding. For an example, see [Image Segmentation Using the Image Segmenter App](#).

Image Batch Processor: Export non-image results using expanded workflows

The Image Batch Processor app now supports more complex workflows. The app can now return other types of information besides a processed image. For an example, see [Batch Processing Using the Image Batch Processor App](#).

imgradient3 and imgradientxyz: Calculate 3-D gradient magnitude, direction, and elevation

The toolbox includes two new functions for computing 3-D image gradients: `imgradient3` and `imgradientxyz`. `imgradient3` computes the gradient magnitude, direction, and elevation. `imgradientxyz` computes the X, Y, and Z directional gradients.

C Code Generation: Generate code from 20 additional functions using MATLAB Coder

The following table lists the Image Processing Toolbox functions that have been enabled for code generation in this release. For all target platforms, these functions generate C code. Some of these functions can also generate C code that uses a precompiled, platform-specific shared library (`.dll`, `.so`, or `.dylib`). Using a shared library preserves performance optimizations in these functions but limits the target to only those platforms that support MATLAB (see system requirements). For a complete list of Image Processing Toolbox functions that support code generation, see [List of Supported Functions with Usage Notes](#).

<code>adaptthresh</code> ¹	<code>imbinarize</code> ¹	<code>imgradientxyz</code>	<code>offsetstrel</code>
<code>boundarymask</code>	<code>imboxfilt</code> ¹	<code>imoverlay</code>	<code>otsuthresh</code> ¹
<code>bwboundaries</code> ¹	<code>imfindcircles</code> ¹	<code>impyramid</code>	<code>rgb2lab</code>
<code>bwconncomp</code>	<code>imgaussfilt</code> ¹	<code>lab2rgb</code>	<code>superpixels</code>
<code>demosaic</code>	<code>imgradient3</code>	<code>label2idx</code>	

¹ If you choose the generic MATLAB Host Computer option in the MATLAB Coder configuration settings, this function generates C code that uses a precompiled, platform-specific shared library.

The function listed in the following table only generates C code that uses a platform-specific shared library. When using this function, choose the generic MATLAB Host Computer option in the MATLAB Coder configuration settings.

imread	
--------	--

The functions listed in the following table previously only generated C code, and can now also generate C code that uses a platform-specific shared library. To use a shared library, choose the generic MATLAB Host Computer option in the MATLAB Coder configuration settings.

rgb2gray	rgb2hsv
----------	---------

In addition, the following function that already supported code generation has additional capabilities.

Function	New Capability
regionprops	Supports the connected components structure for code generation.

imbinarize, otsuthresh, and adapthresh: Threshold images using global and locally adaptive thresholds

The toolbox includes the new function, `imbinarize`, that converts grayscale images to binary images using global threshold or a locally adaptive threshold. The toolbox includes two new functions, `otsuthresh` and `adapthresh`, that provide a way to determine the threshold needed to convert a grayscale image into a binary image. This function replaces the `im2bw` function.

Structuring Elements: New shapes and a new function

The `strel` function, which you use to create structuring elements used in morphological operations, supports several new 3-D shapes: 'sphere', 'cube', and 'cuboid'.

In addition, the toolbox now includes a new function, named `offsetstrel`, that you use to create the nonflat structuring elements, 'ball' and 'arbitrary', formerly created using the `strel` function. You now use the `strel` function to create flat structuring elements, such as 'disk' and 'line'.

DICOM functions support non-ASCII character sets

The DICOM functions `dicominfo` and `dicomwrite` now support non-ASCII character sets. DICOM files can contain non-ASCII data, such as Kanji, Katakana, and Hiragana characters, in personal name fields and other fields.

edge: Change to Canny Edge Detection

The implementation of the Canny method of edge detection in the `edge` function has been modified to make the gradient computation more accurate.

Compatibility Considerations

Because of this change, the `edge` function returns more accurate results; however, the results returned are different than what they were in previous releases.

Performance improvements

The performance of the following functions has improved:

- `rgb2lab`
- `rgb2xyz`

Color space conversion functions: improved precision and numerical consistency

The following color space conversion functions have improved precision and numerical accuracy:

- `lab2rgb`
- `xyz2rgb`
- `rgb2lab`
- `rgb2xyz`

Compatibility Considerations

Because of the improved precision and numerical consistency of the `lab2rgb`, `xyz2rgb`, `rgb2lab`, `rgb2xyz` functions, the results they return are different than what they returned in previous releases.

Functions Being Removed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
<code>corner</code>	Still works	Use <code>detectHarrisFeatures</code> or <code>detectMinEigenFeatures</code> in Computer Vision System Toolbox™ instead.	Replace instances of <code>corner</code> with the appropriate replacement function.
<code>cornermetric</code>	Still works	Use <code>detectHarrisFeatures</code> or <code>detectMinEigenFeatures</code> and the <code>cornerPoints</code> class in Computer Vision System Toolbox instead.	Replace instances of <code>cornermetric</code> with the appropriate replacement function.
<code>im2bw</code>	Still works	Use <code>imbinarize</code> instead.	Replace instances of <code>im2bw</code> with <code>imbinarize</code> .

R2015b

Version: 9.3

New Features

Bug Fixes

Compatibility Considerations

C Code Generation: Generate code from more than 20 functions using MATLAB Coder

The following table lists the Image Processing Toolbox functions that have been enabled for code generation in this release. For all target platforms, these functions generate C code. Some of these functions can also generate C code that uses a precompiled, platform-specific shared library (.dll, .so, or .dylib). Using a shared library preserves performance optimizations in these functions but limits the target to only those platforms that support MATLAB (see system requirements). For a complete list of Image Processing Toolbox that support code generation, see List of Supported Functions with Usage Notes.

bwareaopen	houghpeaks	immse	integralBoxFilter
grayconnected ¹	imabsdiff	imresize	psnr
hough	imcrop	imrotate ¹	
houghlines	imgaborfilt	imtranslate ¹	

¹ If you choose the generic MATLAB Host Computer option in the MATLAB Coder configuration settings, this function generates C code that uses a precompiled, platform-specific shared library.

This release also includes functions, listed in the following table, that previously generated C code that uses a platform-specific shared library but now also generate C code that does not require a shared library. For these functions, if you choose the generic MATLAB Host Computer option in the MATLAB Coder configuration settings, these functions generate C code that uses a precompiled, platform-specific shared library.

histeq	im2uint16	imlincomb	rgb2ycbcr
im2int16	imadjust	intlut	ycbcr2rgb

gabor and imgaborfilt: New class and function for designing and applying Gabor filter banks for use in edge and texture analysis

The toolbox includes two new functions, `imgaborfilt` and `gabor`, that provide an easy-to-use interface for Gabor filtering.

grayconnected and imboxfilt: Segment regions by intensities and apply spatial domain filters

The toolbox includes a new function, `grayconnected`, that provides a way to segment regions of similar intensity in a grayscale image. In addition, the toolbox includes several new functions that create an integral image, `integralImage` and `integralImage3`, and that can use the integral image for filtering, `imboxfilt`, `imboxfilt3`, `integralBoxFilter`, and `integralBoxFilter3`.

Performance: Performance improvements for grayscale morphology and image filtering

The performance of several morphological processing functions and image filtering functions has been improved.

dpxread and dpxinfo: Read Digital Picture Exchange files

The toolbox includes two new functions, `dpxread` and `dpxinfo`, that provide an easy-to-use interface for reading DPX files.

imwarp function supports new SmoothEdges parameter

The `imwarp` function supports a new parameter, `SmoothEdges`, that controls whether `imwarp` pads the input image to create a smoother edge in the output image.

Compatibility Considerations

The default value of `SmoothEdges` is `false`, which produces different results than in previous releases. To obtain the same edge behavior as in previous releases, set `SmoothEdges` to `true`.

DICOM functions support new options

The `dicominfo`, `dicomread`, `dicomdisp`, and `adddicomanon` functions supports a new parameter, `UseVRHeuristic`. When set to `true` (the default), this parameter instructs the parser to use a heuristic to help read certain noncompliant files which switch value representation (VR) modes incorrectly. The functions issue a warning if they use the heuristic.

The `dicominfo` function supports a new parameter, `UseDictionaryVR`, which specifies whether the data types of returned metadata should conform to the data dictionary, regardless of what information is present in the file.

New example shows use of imaging functions with the Vision HDL Toolbox

There is a new example, “Generate HDL Code for Image Sharpening,” that showcases use of Image Processing Toolbox functions with the Vision HDL Toolbox™.

R2015a

Version: 9.2

New Features

Bug Fixes

Compatibility Considerations

C Code Generation: Generate code from more than 20 additional functions, including regionprops, watershed, bweuler, bwlabel, bwperim, and multithresh, using MATLAB Coder

The following table lists the Image Processing Toolbox functions that have been enabled for code generation in this release. For all target platforms, these functions generate C code. If you choose the generic MATLAB Host Computer option in the MATLAB Coder configuration settings, many of these functions also generate C code that uses a precompiled, platform-specific shared library (.dll, .so, or .dylib). Using a shared library preserves performance optimizations in these functions but limits the target to only those platforms that support MATLAB (see system requirements). For a complete list of Image Processing Toolbox that support code generation, see List of Supported Functions with Usage Notes.

bweuler ¹	bwperim ¹	watershed ¹
bwlabel	regionprops ¹	

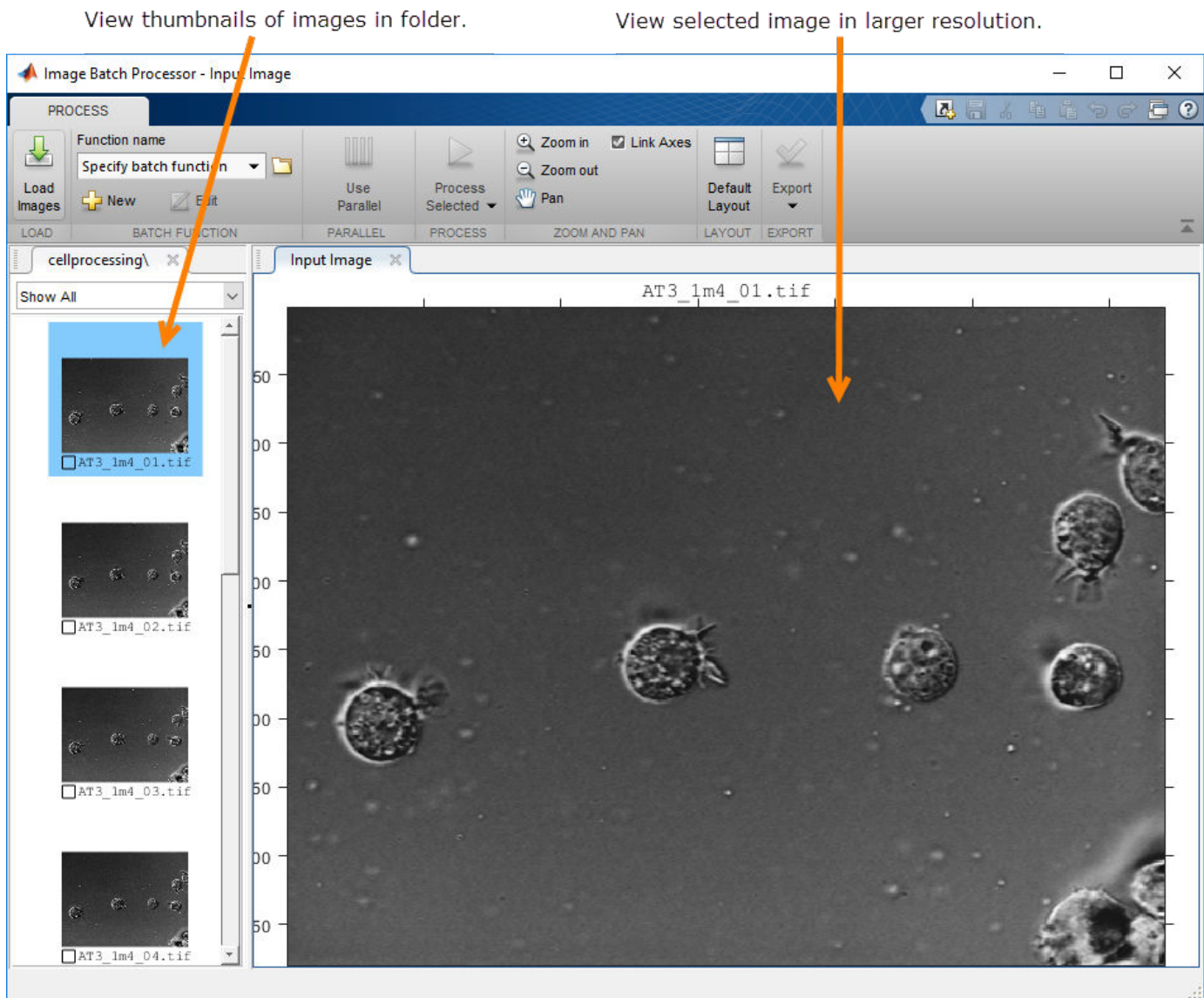
¹ If you choose the generic MATLAB Host Computer option in the MATLAB Coder configuration settings, these functions generate C code that uses a precompiled, platform-specific shared library.

This release also includes functions, listed in the following table, that previously generated C code that uses a platform-specific shared library but now also generate C code that does not require a shared library. You choose which by specifying the platform in MATLAB Coder. If you choose the generic MATLAB Host Computer option in the MATLAB Coder configuration settings, these functions generate C code that uses a precompiled, platform-specific shared library.

bwselect	imclearborder	imfill	imhmin	imregionalmin	multithresh
edge	imextendedmax	imhist	imreconstruct	imwarp	ordfilt2
im2uint8	imextendedmin	imhmax	imregionalmax	medfilt2	stretchlim

App for batch processing sets of images

The toolbox includes new app that facilitates batch processing called the Batch Processing app, shown below. The Batch Processing app enables you to perform an image processing operation on a selection of files or the entire contents of a folder. Using the app, you specify the folder and the function that you want executed. For an example, see Batch Processing Using the Image Batch Processor App.



Fast geodesic interactive segmentation

The toolbox includes a new function, `imseggeodesic`, that provides adaptive, geodesic distance-based binary or trinary segmentation for color images. With this function, you can specify a few pixels, called scribbles, belonging to the different regions (for example, background and foreground in binary segmentation) in the image, and from them the whole image is automatically segmented.

Optimized function for Gaussian filtering

The toolbox includes two new functions, `imgaussfilt` and `imgaussfilt3`, that provide an easy-to-use interface for Gaussian filtering for both 2-D and 3-D data. With these functions, you can switch between filtering in the spatial or frequency domains. In addition, these functions support anisotropic Gaussian filters, enabling you to apply Gaussian smoothing with different standard deviations along

each data dimension. Use these functions instead of using `fspecial` and `imfilter` to create a Gaussian kernel and then applying it.

Fill entire region including border pixels

The toolbox includes a new function, `regionfill`, that provides a way to fill a specified region in an image using inward interpolation so that the region blends in with the surrounding background. `regionfill` replaces the `roifill` function. `regionfill` fill in regions in an image including the border pixels. `roifill` left the pixels on the border of the region unprocessed.

Visualize results of boundary tracing

The toolbox includes a new function, `visboundaries`, that provides a way to plot region boundaries on set of axes. You can use this function to visualize the results returned by the `bwboundaries` function.

Examine contents of DICOM files

The toolbox includes a new function, `dicomdisp`, that displays the metadata of a DICOM file at the command prompt. `dicomdisp` can be useful when debugging issues with DICOM files.

Live image capture in Color Thresholder app

You can now do color thresholding on an image acquired from a Webcam using the Color Thresholder app. The new Image Capture tab allows you to bring a live image 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 Color Thresholder app allows you to:

- Capture live images from USB Webcams
- Save an acquired image to a variable
- Integrate between image acquisition and color thresholding
- Control camera properties, such as brightness and contrast

Use the `colorThresholder` function to open the app. Then select **Load Image > Acquire Image From Camera** to open the Image Capture tab. Select your device, set any properties, and preview the image. You can then capture an image.

regionprops function can return results in table

The `regionprops` function can now return results in a table. To use this capability, specify the string `'table'` as the first input argument.

GPU acceleration for `imregionalmax`, `imregionalmin`, `imgaussfilt`, `imgaussfilt3`, and `regionprops` functions

This release adds GPU acceleration for several toolbox functions: `imgaussfilt`, `imgaussfilt3`, `imregionalmax`, `imregionalmin`, and `regionprops`. GPU acceleration for these functions

requires Parallel Computing Toolbox software and meeting the GPU computing requirements detailed here.

Functions Being Removed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
roifill	Still works but issues a warning	regionfill	Replace instances of roifill with regionfill.

R2014b

Version: 9.1

New Features

Bug Fixes

Apps for image segmentation and region analysis

The toolbox includes new apps:

- Image Segmenter app
- Image Region Analyzer app

Image Segmenter app

The Image Segmenter app enables you to segment images using the active contours (snakes) algorithm. In this app, you first initialize the segmentation by specifying a rough segmentation or initial condition. When you click **Evolve**, the app evolves the initial segmentation, performing the number of iterations you specify, creating a binary mask image. For an example, see [Image Segmentation Using the Image Segmenter App](#).

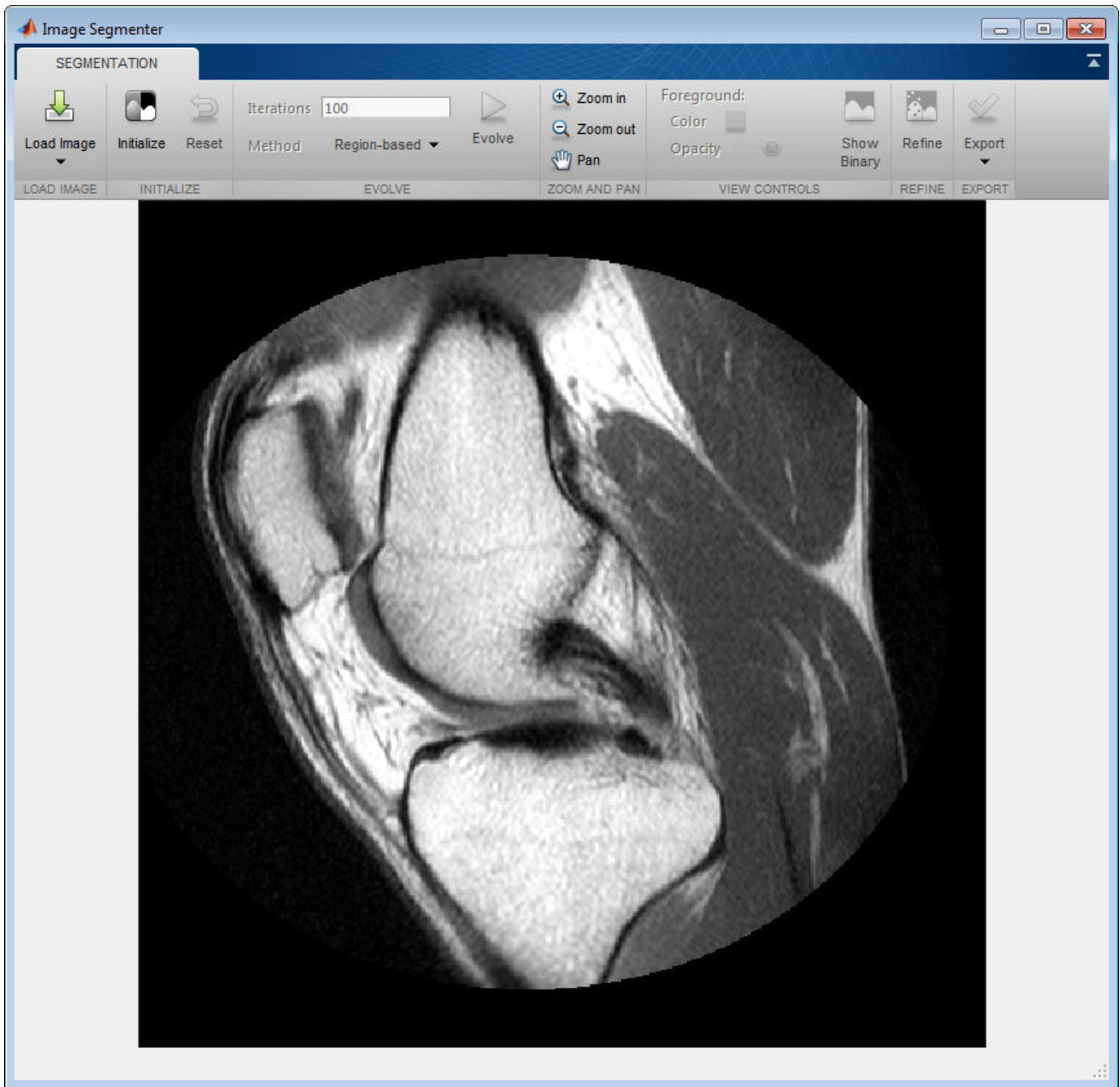
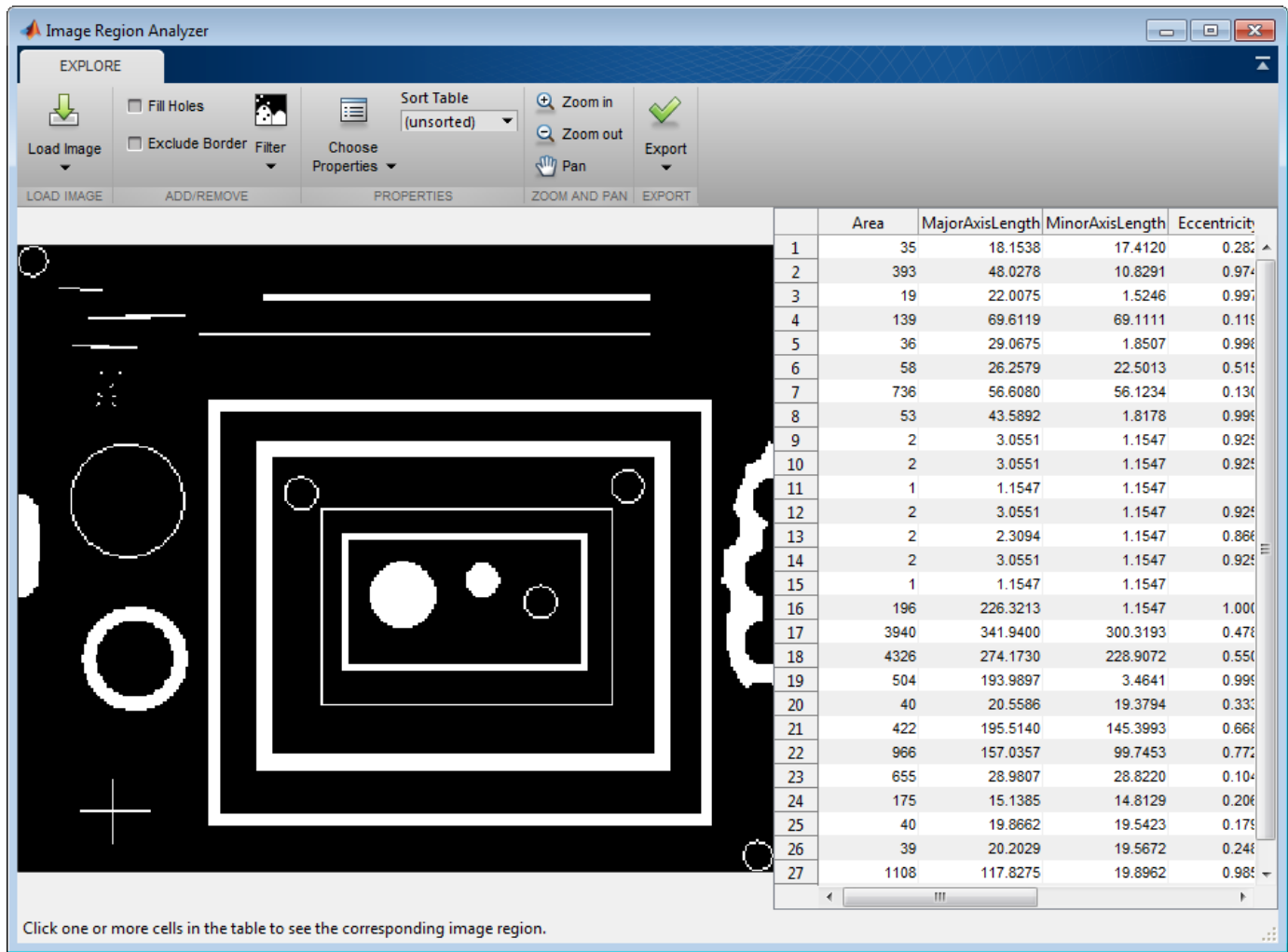


Image Region Analyzer app

The Image Region Analyzer app enables you to explore binary images and filter images based on the properties of regions in the image. For example, you can filter an image to remove all objects smaller than a particular size. The following figure shows the Image Region Analyzer app. When you select the value of a property in the table, the app highlights the corresponding region in the image. For examples, see Calculate Region Properties Using Image Region Analyzer and Filter Images on Region Properties Using Image Region Analyzer App.



C Code Generation: Generate code from 16 additional functions, including `bwtraceboundary`, `imadjust`, `imclearborder`, and `medfilt2`, using MATLAB Coder

This release includes 16 additional toolbox functions that support the generation of C code using MATLAB Coder. Some Image Processing Toolbox functions generate C code that depends on a platform-specific shared library (`.dll`, `.so`, or `.dylib`). Using a shared library preserves performance optimizations in these functions but limits the target to only those platforms that support MATLAB (see system requirements).

The following table lists the Image Processing Toolbox functions that have been enabled for code generation in this release. The table identifies functions that can also generate code that uses a shared library. For a complete list of Image Processing Toolbox that support code generation, see List of Supported Functions with Usage Notes.

<code>bwdist</code> ¹	<code>imadjust</code> ¹	<code>intlut</code> ¹	<code>ordfilt2</code> ¹	<code>ycbcr2rgb</code> ¹
<code>bwtraceboundary</code>	<code>imclearborder</code> ¹	<code>iptcheckmap</code>	<code>rgb2gray</code> ¹	

<code>fitgeotrans</code>	<code>imlincomb</code> ¹	<code>medfilt2</code> ¹	<code>rgb2ycbcr</code> ¹	
<code>histeq</code> ¹	<code>imquantize</code>	<code>multithresh</code> ¹	<code>stretchlim</code> ¹	

¹ Generated code uses a precompiled, platform-specific shared library.

This release also includes nine functions, listed in the following table, that generate C code or generate C code that uses a platform-specific shared library, depending on which platform you specify in MATLAB Coder. If you choose the generic MATLAB Host Computer option in the MATLAB Coder configuration settings, these functions generate C code that uses a precompiled, platform-specific shared library.

<code>bwlookup</code>	<code>imclose</code>	<code>imfilter</code>
<code>bwmorph</code>	<code>imdilate</code>	<code>imopen</code>
<code>imbothat</code>	<code>imerode</code>	<code>imtophat</code>

Nonrigid image registration

The toolbox includes a new function, `imregdemons`, that provides a way to perform nonrigid image registration. The toolbox already supports several types of rigid image registration that work at a global level, applying the same mathematical transformation to every pixel in an image. Now the toolbox supports a function that works at a local level, capable of applying different transformations to every image pixel. The `imregdemons` function returns the warped image and a displacement field that describes how each pixel is transformed.

Image warping using displacement fields

The `imwarp` function now accepts the displacement field returned by the `imregdemons` as input. The `imregdemons` function returns a warped image and a displacement field that describes how each pixel is transformed. If you want more detailed control of a nonrigid registration, you can pass this displacement field to `imwarp` to perform the transformation. For example, by using `imwarp`, you can specify the interpolation method used.

Image segmentation using the Fast Marching Method algorithm

The toolbox includes a new function, `imsegfmm`, that segments an image using the Fast Marching Method (FMM) algorithm. To segment an image, you must first create a weight image using either the `gradientweight` or `graydiffweight` functions. These functions create an image in which every pixel is a value. You then pass this weight image to `imsegfmm`, specifying where the algorithm should start, the seed location.

Image comparison using mean-squared error

The toolbox includes a new function, `immse`, that calculates the mean-squared error.

Image filtering based on object properties

The toolbox includes two new functions, `bwareafilt` and `bwpropfilt` that filter images based on the values of image properties. For example, using `bwareafilt`, you can remove objects from an image if their area is less than a specified number of pixels.

Color space conversion functions

The toolbox includes several new functions to convert between the RGB, XYZ, and L^*a^*b color spaces.

<code>rgb2lab</code>	<code>rgb2xyz</code>	<code>lab2xyz</code>
<code>lab2rgb</code>	<code>xyz2rgb</code>	<code>xyz2lab</code>

Use these new conversion functions instead of using `makecform` with these conversion types: `lab2srgb`, `srgb2lab`, `srgb2xyz`, `xyz2srgb`, `lab2xyz`, and `xyz2lab`.

activecontour function supports parameter to control tendency of contour to expand or contract

The `activecontour` function supports a new parameter, `ContractionBias`, that influences whether the contour grows outward or shrinks inward during segmentation.

Region-of-Interest (ROI) functions now support deletion from context menu

The `imellipse`, `imfreehand`, `imline`, `impoint`, `impoly`, and `imrect` functions, that you use to define regions of interest in images, now support a deletion option from their context menus.

dicomwrite function now supports the ability to specify the bitdepth of images written

The `dicomwrite` function now supports the `UseMetadataBitDepths` parameter which you can use to specify the bitdepth of the image written to the DICOM file.

GPU acceleration for bwlabel and imregdemons

This release adds GPU acceleration for two toolbox functions: `bwlabel` and `imregdemons`. GPU acceleration for these functions requires Parallel Computing Toolbox software and meeting the GPU computing requirements detailed here.

R2014a

Version: 9.0

New Features

Bug Fixes

Compatibility Considerations

C Code Generation for more than 25 functions, including edge, imfilter, imwarp, imopen, imclose, imerode, and imdilate using MATLAB Coder

You can generate standalone C code for a group of toolbox functions, listed below. Generating code requires MATLAB Coder.

affine2d	im2double	imclose	imhist	mean2
bwpack	im2int16	imdilate	imopen	projective2d
bwselect	im2single	imerode	imref2d	strel
bwunpack	im2uint16	imextendedmax	imref3d	
edge	im2uint8	imextendedmin	imtophat	
getrangefromclass	imbothat	imfilter	imwarp	

GPU acceleration for an additional nine functions, including bwdist, imfill, imreconstruct, iradon, radon, and stretchlim

This release adds GPU acceleration for an additional nine toolbox functions, listed below. GPU acceleration for these functions requires Parallel Computing Toolbox software and meeting the GPU computing requirements detailed here.

bwdist	imreconstruct	normxcorr2
imcomplement	iradon	radon
imfill	mean2	stretchlim

App for color image thresholding

The toolbox includes a new Color Thresholder app that enables you to segment color images by manipulating their color components. The app presents the image using several standard color spaces. You choose which color space representation gives the best contrast between foreground and background and then use the component histograms to segment the image.

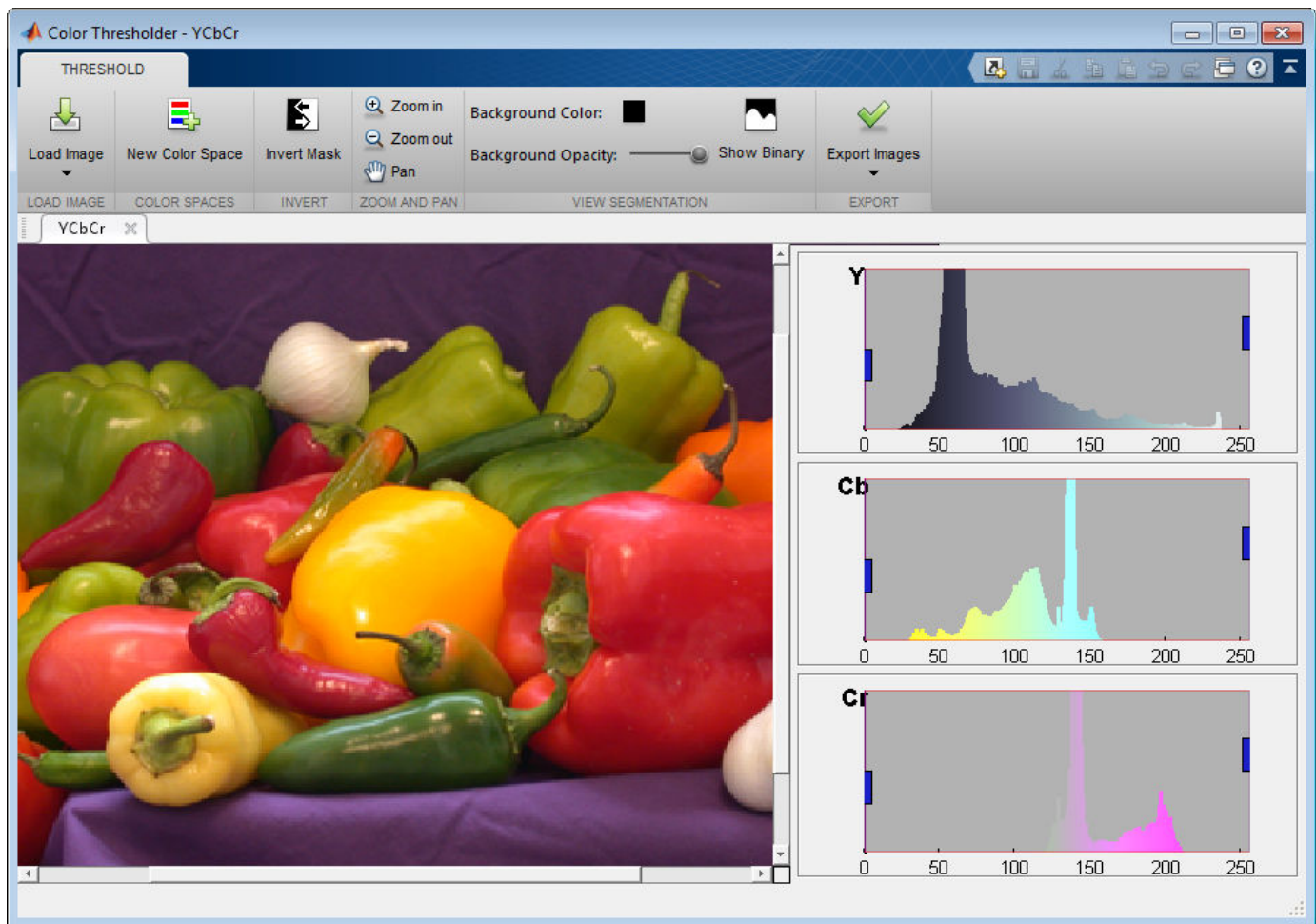


Image quality metrics, including peak signal to noise (psnr) and structured similarity metric (ssim)

The toolbox includes two new functions, `ssim` and `psnr`, that compute image quality metrics.

Guided filtering for image enhancement

The toolbox includes a new function, `imguidedfilter`, that provides an edge-preserving nonlinear filter for use with images.

Phase correlation and translation-only image registration functions

The toolbox includes a new function, `imregcorr`, that applies an FFT-based transform to register two images with regard to translation, rotation, and scale. The toolbox includes a new function, `imtranslate`, that applies a translation transformation to an input image and returns the transformed image. Using `imtranslate`, you can provide spatial reference information, specify the method used for interpolation, and control other aspects of the translation.

File names of Image Processing Toolbox examples changed

The file names of Image Processing Toolbox examples have changed. The following table lists the example files in alphabetical order by their former names.

Old Example File Name	New Example File Name
ipexaerial.m	RegisterAerialPhotoExample.m
ipexangle.m	MeasureAngleExample.m
ipexautorotate.m	RotationFeatureMatchingExample.m
ipexbatch.m	BatchProcessImageExample.m
ipexblind.m	BlindImageDeblurringExample.m
ipexblockprocedge.m	BlockProcessLargeImageExample.m
ipexblockprocstats.m	BlockProcessStatisticsExample.m
ipexcell.m	CellSegmentationExample.m
ipexcheckerboard.m	GalleryTransformedImagesExample.m
ipexcircles.m	DetectCirclesExample.m
ipexconformal.m	ConformalMappingImageExample.m
ipexcontrast.m	ContrastEnhancementExample.m
ipexfabric.m	LabColorSegmentationExample.m
ipexhistology.m	KMeansSegmentationExample.m
ipexknee.m	RegisterMultimodalImagesExample.m
ipexlanstretch.m	MultispectralImageEnhancementExample.m
ipexlucy.m	LucyRichardsonImageDeblurringExample.m
ipexmri.m	MRISliceExample.m
ipexndvi.m	MultispectralVegetationSegmentationExample.m
ipexnormxcorr2.m	RegisterNormalizedCrossCorrelationExample.m
ipexpendulum.m	PendulumLengthExample.m
ipexprops.m	MeasureGrayscaleRegionsExample.m
ipexradius.m	MeasureRadiusExample.m
ipexreconstruct.m	ReconstructImageExample.m
ipexregularized.m	RegularizedImageDeblurringExample.m
ipexrice.m	NonuniformIlluminationExample.m
ipexrotate.m	RotationFitgeotransExample.m
ipexroundness.m	RoundObjectsExample.m
ipexshear.m	PadShearImageExample.m
ipexsnow.m	SnowflakesGranulometryExample.m
ipextexturefilter.m	TextureSegmentationExample.m
ipextraffic.m	TrafficSegmentationExample.m
ipexwatershed.m	WatershedSegmentationExample.m

Old Example File Name	New Example File Name
ipexwiener.m	WienerImageDeblurringExample.m

Location of sample images changed

The location of Image Processing Toolbox sample images has changed. These images were stored in the `imdemos` folder and are now stored in the `imdata` folder.

regionprops function uses new algorithm to calculate perimeter

The `regionprops` function uses a new algorithm to calculate a perimeter, when used with the `'Perimeter'` option. Because of this change, `regionprops` returns different results for the perimeter calculations than it did in earlier releases.

Compatibility Considerations

While the new algorithm used with `regionprops` returns more accurate perimeter calculation, you can get the same return value as previous releases by specifying the `'perimeterold'` option.

R2013b

Version: 8.3

New Features

Bug Fixes

Compatibility Considerations

GPU acceleration for more than 20 functions, including `bwmorph`, `edge`, `imresize`, and `medfilt2`

This release introduces GPU acceleration for a group of toolbox functions, listed below. GPU acceleration for these functions requires Parallel Computing Toolbox software and meeting the GPU computing requirements detailed here.

<code>bwmorph</code>	<code>im2single</code>	<code>imgradientxy</code>	<code>medfilt2</code>
<code>corr2</code>	<code>im2uint8</code>	<code>imhist</code>	<code>rgb2gray</code>
<code>edge</code>	<code>im2uint16</code>	<code>imlincomb</code>	<code>rgb2ycbcr</code>
<code>histeq</code>	<code>imabsdiff</code>	<code>imnoise</code>	<code>std2</code>
<code>im2double</code>	<code>imadjust</code>	<code>imresize</code>	<code>stdfilt</code>
<code>im2int16</code>	<code>imgradient</code>	<code>mat2gray</code>	<code>ycbcr2rgb</code>

Additional 2D geometric transformations: piecewise linear, local weighted mean, and polynomial

The toolbox includes several new classes for representing 2D geometric transformations, listed below.

<code>images.geotrans.PiecewiseLinearTransformation2d</code>	2D piecewise linear geometric transformation
<code>images.geotrans.PolynomialTransformation2d</code>	2D polynomial geometric transformation
<code>images.geotrans.LocalWeightedMeanTransformation2d</code>	2D local weighted mean geometric transformation

Additional parameter in `imregister` and `imregtform` to specify initial transformation

The `imregister` and `imregtform` functions both support a new parameter, `InitialTransformation`, that enables you to specify the starting point for the transformation. Specifying an initial geometric transformation as a starting point, the functions can return better results.

`fitgeotrans` function for fitting geometric transformation to control point pairs

The toolbox includes a new function, `fitgeotrans`, that takes pairs of control points and uses them to infer a geometric transformation. The function returns a geometric transformation object that can be used with `imwarp`.

`imregister` and `imregtform` Return Different Values

Some changes to the implementation of the `imregister` and `imregtform` functions might cause these functions to return different results than they did before R2013b.

Compatibility Considerations

You might need to change parameters to the `imregister` and `imregtform` functions to achieve similar results to what you had previously.

Functions Being Removed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
<code>cp2tform</code>	Still works but issues a warning	<code>fitgeotrans</code>	Replace instances of <code>cp2tform</code> with <code>fitgeotrans</code> .
<code>imtransform</code>	Still works but issues a warning	<code>imwarp</code>	Replace instances of <code>imtransform</code> with <code>imwarp</code> .
<code>maketform</code>	Still works but issues a warning	<code>fitgeotrans</code> , <code>affine2d</code> , <code>affine3d</code> , or <code>projective2d</code>	Replace instances of <code>maketform</code> with one of the recommended functions.

R2013a

Version: 8.2

New Features

Bug Fixes

Image segmentation using active contours

The toolbox includes a new function, `activecontour`, for segmenting an image using the active contours (snakes) algorithm.

Classes and functions for representing and applying 2-D and 3-D geometric transformations

The toolbox includes several new classes for representing 2-D and 3-D geometric transformations, listed below.

<code>affine2d</code>	2-D affine geometric transformation
<code>affine3d</code>	3-D affine geometric transformation
<code>projective2d</code>	2-D projective geometric transformation
<code>imregtform</code>	Estimate the geometric transformation that aligns two images
<code>imwarp</code>	Apply geometric transformation to image

Classes for defining world coordinate system of an image

The toolbox includes several new classes for representing world coordinate systems associated with an image, listed below.

<code>imref2d</code>	Reference 2-D image to world coordinates
<code>imref3d</code>	Reference 3-D image to world coordinates

Code generation for `conndef`, `imcomplement`, `imfill`, `imhmax`, `imhmin`, `imreconstruct`, `imregionalmax`, `imregionalmin`, `iptcheckconn`, and `padarray` functions (using MATLAB Coder)

You can generate standalone C code for a group of toolbox functions, listed below. Generating code requires MATLAB Coder.

<code>conndef</code>	<code>imcomplement</code>
<code>imfill</code>	<code>imhmax</code>
<code>imhmin</code>	<code>imreconstruct</code>
<code>imregionalmax</code>	<code>imregionalmin</code>
<code>iptcheckconn</code>	<code>padarray</code>

GPU acceleration for `imrotate`, `imfilter`, `imdilate`, `imerode`, `imopen`, `imclose`, `imtophat`, `imbothat`, `imshow`, `padarray`, and `bwlookup` functions (using Parallel Computing Toolbox)

This release introduces GPU acceleration for a group of toolbox functions, listed below. GPU acceleration for these functions requires Parallel Computing Toolbox software and meeting the GPU computing requirements detailed here.

<code>bwlookup</code>	<code>imbothat</code>
<code>imclose</code>	<code>imdilate</code>
<code>imerode</code>	<code>imfilter</code>
<code>imopen</code>	<code>imrotate</code>
<code>imshow</code>	<code>imtophat</code>
<code>padarray</code>	<code>std2</code>

Unsharp mask filtering

The toolbox includes a new function, `imsharpen`, that does unsharp mask filtering. Use this new function instead of using the 'unsharp' option with the `fspecial` function.

R2012b

Version: 8.1

New Features

Bug Fixes

Compatibility Considerations

Image gradient computation with `imgradient` and `imgradientxy` functions

The toolbox includes two new functions for computing image gradients: `imgradient` and `imgradientxy`. `imgradient` computes the gradient magnitude and direction. `imgradientxy` computes the X and Y directional gradients

Histogram matching with `imhistmatch` function

The new function `imhistmatch` adjusts the histogram of an image to match the N-bin histogram of a reference image.

Multilevel thresholding with `multithresh` and `imquantize` functions

The new `imquantize` and `multithresh` functions enable multilevel grayscale thresholding and labeling. `imquantize` labels an image using a fixed range of grayscale levels. `multithresh` computes threshold levels using Otsu's method. The levels computed by `multithresh` can be used as input to `imquantize`.

3-D image registration with `imregister` function

In addition to 2-D images, `imregister` can now align 3-D images using intensity-based registration of the images' voxel data.

Code generation for `bwmorph` and `bwlookup` with MATLAB Coder

Standalone C code can be generated for `bwmorph` and `bwlookup`. The generated C code meets the strict memory and data type requirements of embedded target environments. To generate this code you need a MATLAB Coder license.

Added function `bwlookup`

Compatibility Considerations

Function `applylut` is not recommended; use `bwlookup` instead.

Writing private metadata when anonymizing DICOM files

The `dicomanon` function now supports writing private metadata fields using the 'WritePrivate' parameter

Expanded color options with `imshowpair`

`imshowpair` now supports additional color options for displaying image differences and stereo imagery when using the new 'ColorChannels' parameter. Using the 'red-cyan' value with this parameter is particularly useful for viewing stereo anaglyphs.

Performance improvements

The performance of the following functions has improved by taking advantage of hardware optimizations and multicore capabilities:

- `adapthisteq`
- `histeq`
- `imrotate`
- `intlut`

The `dicomanon`, `dicominfo`, `dicomlookup`, `dicomread`, `dicomwrite` functions have optimized implementations.

New Example

Detect and measure circular objects in an image (`ipexcircles`).

R2012a

Version: 8.0

New Features

Bug Fixes

Compatibility Considerations

Intensity-Based Image Registration

The new `imregister` function lets you automatically align two images using intensity values, even when the images were created by two different devices (`multimodal`). With intensity-based registration, you do not need to specify control points.

You use the new `imregconfig` function to create the optimizer and the metric that `imregister` uses to specify the desired registration parameters.

Two New Functions to Visually Compare Images

The toolbox includes two new functions for visually comparing images: `imshowpair` and `imfuse`.

`imshowpair` creates a composite of two images and displays them in a figure.

`imfuse` creates a composite of two images and returns a third image that is a numeric matrix containing a fused version of the original images.

Circle Detection Using the Circular Hough Transform

The new `imfindcircles` function uses the Hough transform to find circular elements in grayscale, RGB, or binary images. To view the circles that have been detected, overlaid on the original image, use the `viscircles` function.

Performance Improvements

The performance of the `imlincomb` function has improved by taking advantage of multicore capabilities.

New and Updated Demos

The toolbox includes these new and updated demos.

- Registering Multimodal MRI Images (`ipexknee`)
- Finding the Rotation and Scale of a Distorted Image (`ipexrotate`)
- Measuring the Radius of a Roll of Tape (`ipexradius`) - Updated to use the new `imfindcircles` function

Data Type Change to Output Variable for `bwdist`

The `bwdist` function returns two output variables: the Euclidean distance transform and the closest-pixel map. The data type of the second output variable, the closest-pixel map, has changed.

Compatibility Considerations

Before R2012a, the data type of the second output variable returned by the `bwdist` function was `single`. Now the data type of the second output variable returned by `bwdist` is dynamically chosen.

iradon Function Updated

A small imprecision in the way the dc level was computed in `iradon` has been corrected.

Compatibility Considerations

Output variables I and H will be slightly different (about 1% to 2%) than in previous versions.

Functions Being Removed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
<code>iptcheckinput</code>	Still works but issues a warning	<code>validateattributes</code>	Replace all existing instances of <code>iptcheckinput</code> with <code>validateattributes</code> .
<code>iptcheckstrs</code>	Still works but issues a warning	<code>validatestring</code>	Replace all existing instances of <code>iptcheckstrs</code> with <code>validatestring</code> .
<code>iptchecknargin</code>	Still works but issues a warning	<code>narginchk</code>	Replace all existing instances of <code>iptchecknargin</code> with <code>narginchk</code> .

R2011b

Version: 7.3

New Features

Bug Fixes

Compatibility Considerations

Parallel Block Processing Now Possible with `blockproc`

If you have Parallel Computing Toolbox, you can now use a new option in `blockproc` to improve performance of block processing tasks. Set the `'UseParallel'` argument to `true` to use this option.

New `bwdistgeodesic` Function Computes Geodesic Distance Transform

Use `bwdistgeodesic` to compute the geodesic distance transform for a binary image.

New `graydist` Function Computes Gray-Weighted Distance Transform

Use `graydist` to compute the gray-weighted distance transform of a grayscale image.

New `imapplymatrix` Function Computes Linear Combination of Color Channels

The new `imapplymatrix` function applies a weighted sum to the color planes of an image for use in color space conversions.

Performance Improvements

Faster Functions

- `imhist` for large images
- `rgb2gray`

`hdrread` Now Corrects for Small Values

The `hdrread` function now returns correct small values. Specifically, the special case value of `(0,0,0,0)` now maps to `(0,0,0)`.

Compatibility Considerations

Before R2011b, the value `(0,0,0,0)` mapped to `(0.5740E-41, 0.5740E-41, 0.5740E-41)`.

Change in Behavior for `dicomwrite`

If you use the `dicomwrite` function with the `'CreateMode'` option set to `'Create'` and pass in a data structure that contains the `'InversionTime'` tag, you will always receive `'InversionTime'` in your output.

Compatibility Considerations

Before R2011b, if your input structure contained the `'InversionTime'` field, the DICOM file may or may not contain the `'InversionTime'` field. The inclusion of `'InversionTime'` depended on other parameters.

Warning and Error ID Changes

Many warning and error IDs have changed from their previous versions. These warnings or errors typically appear during a function call.

Compatibility Considerations

If using warning or error IDs, you might need to change the strings you use. For example, if you turned off a warning for a certain ID, the warning might now appear under a different ID. If you use a `try/catch` statement in your code, replace the old identifier with the new identifier. There is no definitive list of the differences, or of the IDs that changed.

Functions and Function Elements Being Removed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
edge function — K-direction syntax	Errors	<code>BW = edge(..., direction)</code>	The syntax <code>BW = edge(...,K)</code> has been removed. Use the <code>BW = edge(...,direction)</code> syntax instead.
edge function — Marr-Hildreth syntax	Errors	<code>edge(I, 'marr-hildreth', ...)</code>	The syntax <code>edge(I, 'marr-hildreth', ...)</code> has been removed. Use the <code>edge(I, 'log', ...)</code> syntax instead.
<code>imfeature</code>	Errors	<code>regionprops</code>	<code>imfeature</code> has been removed. Use <code>regionprops</code> instead.
<code>immovie</code> — <code>immovie(D,size)</code>	Errors	<code>immovie(X,map)</code>	<code>immovie(D,size)</code> is an obsolete syntax and is no longer supported. Use <code>immovie(X,map)</code> instead.
<code>imrotate</code> function — <code>[R,G,B]</code> output syntax	Errors	<code>RGB2 = imrotate(RGB1)</code>	The syntax <code>[R,G,B] = imrotate(RGB)</code> has been removed. Use the <code>RGB2 = imrotate(RGB1)</code> syntax instead.

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
<code>imrotate</code> — no output argument syntax	Starting in R2011b, no output argument syntax returns result in <code>ans</code> .	Call <code>imshow</code> to display the output of <code>imrotate</code>	Replacement is to call <code>imshow</code> to display the output of <code>imrotate</code> , like this: <pre>B = imrotate(A,30); imshow(B)</pre>
<code>imshow</code> function — <code>imshow(..., DISPLAY_OPTION)</code>	Errors	<code>imshow(..., 'InitialMagnification', 100)</code> or <code>imshow(..., 'InitialMagnification', 'fit')</code>	<ul style="list-style-type: none"> The syntax <code>imshow(..., 'true size')</code> has been removed. Use the <code>imshow(..., 'InitialMagnification', 100)</code> syntax instead. The syntax <code>imshow(..., 'notruesize')</code> has been removed. Use the <code>imshow(..., 'InitialMagnification', 'fit')</code> syntax instead.
<code>imshow</code> function — <code>imshow(x,y,...)</code> syntax	Errors	<code>imshow(..., 'XData', x, 'YData', y)</code>	The syntax <code>imshow(x,y,...)</code> has been removed. Use the <code>imshow(..., 'XData', x, 'YData', y)</code> syntax instead.
<code>imshow</code> function — <code>imshow(I,N)</code> syntax	Errors	Not applicable	The syntax <code>imshow(I,N)</code> has been removed. Your grayscale image will be displayed using 256 shades of gray.
<code>imview</code>	Errors	<code>imtool</code>	<code>imview</code> has been removed. Use <code>imtool</code> instead.
<code>iptsetpref</code> function — 'ImshowTruesize' preference	Errors	'ImshowInitialMagnification' preference	Replace all existing instances of 'ImshowTruesize' with 'ImshowInitialMagnification'.

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
iptsetpref function — 'ImviewInitialMagnification' preference	Errors	'ImtoolInitialMagnification' preference	Replace all existing instances of 'ImviewInitialMagnification' with 'ImtoolInitialMagnification'.
Any calls to iptsetpref from within a startup.m file	It will do nothing.	Any calls to iptsetpref in any session are persistent.	You can safely remove calls to iptsetpref from startup.m files.
isbw	Errors	Not applicable	isbw has been removed. No replacement.
isgray	Errors	Not applicable	isgray has been removed. No replacement.
isind	Errors	Not applicable	isind has been removed. No replacement.
isrgb	Errors	Not applicable	isrgb has been removed. No replacement.
medfilt2 function — medfilt2(A,[M N],[Mb Nb],...) syntax	Errors	Not applicable	No replacement.
montage — montage(D,[M N P]) syntax	Errors	Not applicable	Syntax has been removed. No replacement.
uintlut	Errors	intlut	uintlut has been removed. Use intlut instead.
wiener2 function — wiener2(I,[m n],[mblock nblock],...) syntax	Errors	wiener2(I,[m n]) or wiener2(I,[m n],noise)	The syntax wiener2(I,[m n],[mblock nblock]) has been removed. Use the wiener2(I,[m n]) syntax instead. The syntax wiener2(I,[m n],[mblock nblock],noise) has been removed. Use the wiener2(I,[m n],noise) syntax instead.

R2011a

Version: 7.2

New Features

Bug Fixes

Compatibility Considerations

New bwconvhull Function Computes Convex Hull Image

Use `bwconvhull` to compute the convex hull image from a binary image.

New dicomwrite Option Writes Multiframe Imagery to Single File

Set the new 'MultiframeSingleFile' option of the `dicomwrite` function to `true` to write multiframe imagery to one file, regardless of how many frames the input image contains.

nitfread Now Reads NITF Files with JPEG-Compressed Images

If you have NITF files containing JPEG-compressed images, you can now read them using `nitfread`.

Reduced Memory Use for std2

The `std2` function has improved performance for large 1-, 8-, and 16-bit integers.

Compatibility Considerations

Compared to releases before R2011a, the `std2` function now returns slightly different results for some images. To receive the same results as previously, use this code:

```
% For input image im
if ~isa(im,'double')
    im = double(im);
end
std_old = std(im(:));
```

Reduced Memory Use for watershed

The `watershed` function is now more memory efficient than it was in releases before R2011a.

Compatibility Considerations

The `watershed` regions in the label matrix returned by `watershed` have different indices than they did before R2011a.

Also, the label matrix returned by `watershed` was class `double` in previous releases, and is now an unsigned integer class.

If you want to return a label matrix of class `double`, as you did before, use the `double` function to convert it:

```
L = watershed(A);
L = double(L);
```

iccread and iccwrite Now Warn in Cases of Unrecognized PrimaryPlatform Signatures

If `iccread` or `iccwrite` encounter an unrecognized `PrimaryPlatform` signature in the profile header, they will warn. In releases before R2011a, these functions would error instead of warn in cases with unrecognized `PrimaryPlatform` signatures.

Plot Selector Now Includes `implay`

The `implay` function has been added to the list of functions available in the Plot Selector. You can now display data in `implay` directly from the Plot Selector workspace tool. For details about the Plot Selector, see *Enhanced Plot Selector Simplifies Data Display*.

Support for Code Generation from MATLAB

You can now generate standalone C code for two Image Processing Toolbox functions: `label2rgb` and `fspecial`. The generated C code meets the strict memory and data type requirements of embedded target environments. To generate this code you need a MATLAB Coder license. See the *Code Generation for Image Processing Toolbox Functions* chapter in the *User's Guide* for details, including limitations.

edge Function No Longer Smooths Image Twice

In previous releases, the implementation of the Canny filter, called with the `canny` method of the `edge` function, smoothed the image twice while constructing the gradient image. The function smoothed the image once using a Gaussian filter and then used a first derivative of a Gaussian filter to extract a smoothed version of the image gradient. Smoothing an image and then differentiating it is the same as convolving the image with a derivative of the smoothing kernel, so this implementation had the effect of smoothing the image twice. In addition, the original implementation of the Canny filter included an extra morphological thinning step that is not in the published algorithm.

Compatibility Considerations

The `edge` function no longer smooths an image twice. If you are setting the value of `sigma` and want similar results to the previous implementation, increase `sigma` by a factor of `sqrt(2)`.

To achieve the same results produced by the previous implementation, use this syntax:

```
BW = edge(I, 'canny_old', ...)
```

Functions and Function Elements Being Removed

Function or Function Element	What Happens When You Use This Function or Element?	Use This Instead	Compatibility Considerations
<code>ipttable</code>	Errors	<code>uitable</code>	Replace all existing instances of <code>ipttable</code> with <code>uitable</code> .

R2010b

Version: 7.1

New Features

Bug Fixes

Compatibility Considerations

New corner Function Detects Corners in Image

The new `corner` function detects corners in a grayscale or binary image. Corners are a feature you can use to find the correspondence between images.

Compatibility Considerations

In R2008b and later releases, you could find corners by computing a `cornermetric` matrix with the `cornermetric` function and then finding peak values. Now, you can simplify your workflow by using the `corner` function.

Efficient Display and Navigation of Very Large Images of Arbitrary Format in `imtool`

The `rsetwrite` function allows you to create a multiresolution image pyramid (R-Set) from a large image file. In previous releases, the `rsetwrite` function accepted TIFF or NITF image files. Now, in addition to accepting these image files, `rsetwrite` accepts `ImageAdapter` objects. `ImageAdapter` objects allow you to work with images of arbitrary file format. See *Working with Data in Unsupported Formats* for guidelines on how to create an `ImageAdapter` object.

Now Possible to Control Padding Behavior when Using the `blockproc` Function

With the new `'PadMethod'` option, you can now control padding behavior when using the `blockproc` function. In previous releases, the `blockproc` function only supported zero padding along the boundary of the image. Now, the function supports padding the image with a scalar `pad` value, repeated border elements of `A`, or the mirror reflection of `A`.

Writing to JPEG2000 File Format Supported by `blockproc`

The `blockproc` function now provides more flexibility in format choices. The function supports writing to JPEG2000 file formats with `*.jp2`, `*j2c`, or `*.j2k` extensions.

Enhancements to the `dicomread` Function

The `dicomread` function has been enhanced in two ways: the function now reads multiframe imagery faster, and it reads files containing JPEG-2000 encoded imagery.

The `ImageMagnification` Field of the `nitfinfo` Function Now Returns a Numeric Value

The `ImageMagnification` field of the `nitfinfo` function has been updated. Previously, if you used the function to return a structure with file-level metadata, the `ImageMagnification` field of the structure contained an incorrect value. (The incorrect value was either an empty image magnification value or the text value for the field.) Now, the `ImageMagnification` field returns the value for the image magnification.

Performance Improvements

Faster Functions

- `blockproc`
- `imresize`
- `iradon`

Functions and Function Elements Being Removed

Function or Function Element Name	What Happens When You Use This Function or Element?	Use This Instead	Compatibility Considerations
<code>blkproc</code>	Still runs	<code>blockproc</code>	None

R2010a

Version: 7.0

New Features

Bug Fixes

Compatibility Considerations

New ImageAdapter Class Supports Custom File Formats for blockproc

The `blockproc` function, introduced in R2009b, supported file-based block processing for arbitrarily large images. In R2009b, you could use `blockproc` to read or write TIFF images or to read JPEG2000 images. Now, with the addition of the new `ImageAdapter` class, you can design your own class to use `blockproc` with images of arbitrary file format.

The blockproc Function Now Supports Spatially Varying Operations

Additional fields have been added to the `blockproc` “block struct” that contain spatial information. These new fields facilitate operations that depend on location.

Plot Selector Now Generates Plots for imshow and imtool

The Plot Selector workspace tool creates graphs of workspace variables. The `imshow` and `imtool` functions have been added to the list of possible plotting functions available in the Plot Selector. For more information about the Plot Selector, see [Enhanced Plot Selector Simplifies Data Display](#).

makecform Now Supports White Point Adaptation

`makecform` uses the white point specified by the International Color Consortium (ICC) as the default for the `srgb2lab` and `lab2srgb` transform types. You can now adapt to a white point other than `whitepoint('ICC')`, the default value, by using a new syntax to specify the adapted white point:

```
C = makecform(type, 'AdaptedWhitePoint', WP)
```

You can also create a linear chromatic-adaptation transform:

```
C = makecform('adapt', 'WhiteStart', WPS, 'WhiteEnd', WPE, ...  
    'AdaptModel', modelname)
```

This transform allows you to adapt XYZ color values from one white point to another.

Intel Integrated Performance Primitives Library Support Extended to imdilata, imerode, and medfilt2

The functions `imdilate` and `imerode` are now hardware optimized for ones (3) neighborhoods for `single`, `uint8`, and `uint16` input images.

The `medfilt2` function is now hardware optimized for integer data types (`uint8`, `uint16`, and `int16`) and the `single` data type with kernel size 3 x 3.

imreconstruct Now Supports int64 and uint64

The `imreconstruct` function now supports data types `int64` and `uint64`.

Performance Improvements

Faster Functions

- `edge`
- `imdilate`
- `imerode`
- `imfilter`
- `imresize`
- `iradon`
- `medfilt2`

Multithreaded Functions

- `bwmorph`
- `edge`
- `imabsdiff`
- `imadd`
- `imclose`
- `imdivide`
- `immultiply`
- `imopen`
- `iradon`
- `medfilt2`

Non-interactive Syntax of `improfile` Returns Different Output

One of the non-interactive syntaxes of `improfile` now returns different output. The output for the syntax

```
C = improfile(I,xi,yi,N)
```

has changed. In the syntax above, `N` specifies the number of points for which to compute intensity values and `xi` and `yi` specify the spatial coordinates of the endpoints of the line segments.

For a given line defined by `xi` and `yi`, `improfile` now returns a profile sampled at both endpoints and all sampling points in between at roughly unit interval spacing. If the distance between `xi` and `yi` is `N` pixels, the profile is evaluated at `N+1` points.

Compatibility Considerations

In previous releases, if you supplied the `xi` and `yi` end points as (1,1) and (10,1), the profile would be evaluated at nine points, the nine unit-length intervals between 1 and 10 in the continuous x-y plane. These nine points would be the two end points, plus seven points in between.

R2009b

Version: 6.4

New Features

Bug Fixes

Compatibility Considerations

New blockproc Function to Process Large Images

The new `blockproc` function supports file-based block processing for arbitrarily large TIFF images. The new function supports in-memory operations as well as file-to-file processing of images which are too large to load completely into memory.

Compatibility Considerations

In previous releases, you could use the `blkproc` function for in-memory block-processing of images. The `blkproc` function will be removed in a future release. Replace all instances of `blkproc` with `blockproc`.

When updating your code from `blkproc` to `blockproc`, it is important to note that the user-defined function, `fun`, has a new signature. It now takes a structure, the "block struct," as input instead of simply a matrix of image data.

The example below demonstrate how to update your code from `blkproc` to `blockproc`.

```
% BLKPROC code
I = imread('cameraman.tif');
fun = @dct2;
J = blkproc(I,[8 8],fun);

% BLOCKPROC equivalent (using an anonymous function)
fun = @(block_struct) dct2(block_struct.data);
J = blockproc(I,[8 8],fun);
```

Here's another example showing how to update your code from `blkproc` to `blockproc`.

```
% BLKPROC code
I = imread('concordorthophoto.png');
h = fspecial('gaussian',[11 11],2.5);
fun = @(x) imfilter(x,h,'conv','same');
J = blkproc(I,[500 500],[5 5],fun);

% BLOCKPROC equivalent (using an anonymous function)
fun = @(block_struct) imfilter(block_struct.data,h,'conv','same');
J = blockproc(I,[500 500],fun,'BorderSize',[5 5]);
```

Intel Integrated Performance Primitives Library Upgraded and Support Extended to maci64

The Intel Integrated Performance Primitives (Intel IPP) Library has been upgraded from Version 5.3.1 to Version 6.0 Update 1. Intel IPP Library support has been extended to 64-bit Intel-based Mac computers.

Expanded hough Function Allows Specification of Arbitrary Theta Search Space

The `hough` function now yields faster results for narrower *theta* ranges due to the addition of a parameter/value pair for specifying *theta* values.

Compatibility Considerations

In previous releases, the 'ThetaResolution' parameter controlled the *theta* values for the hough function. Now 'ThetaResolution' is being replaced by the new 'Theta' parameter.

Function Elements Being Removed

Function and Syntax	What Happens When You Use the Function or Element?	Use This Instead	Compatibility Considerations
hough(BW, 'ThetaResolution', val)	Still runs	hough(BW, 'Theta', -90:val:(90-val))	Input parameter no longer recommended. Use new 'Theta' parameter.

Note that with the introduction of the 'Theta' parameter, not all abbreviated forms of 'ThetaResolution' will still work. In previous releases, if you entered the following syntax:

```
hough(BW, 'T', val)
```

'T' stood for 'ThetaResolution'. Now if you enter this same syntax, 'T' stands for the new 'Theta' parameter.

If you have old code that uses the 'ThetaResolution' parameter, please see the definition below:

Parameter	Description
'ThetaResolution'	Real scalar value between 0 and 90, exclusive, that specifies the spacing (in degrees) of the Hough transform bins along the <i>theta</i> axis. Default: 1.

For 'ThetaResolution', $ntheta = 2 * \text{ceil}(90 / \text{ThetaResolution})$. *theta* angle values are in the range [-90, 90) degrees. If $90 / \text{ThetaResolution}$ is not an integer, the actual angle spacing is $90 / \text{ceil}(90 / \text{ThetaResolution})$.

The imfilter Function Now Faster for uint16 and double Inputs

The `imfilter` function now runs faster with `uint16` and `double` inputs than in previous releases. This performance enhancement is due to the use of the Intel IPP Library with inputs of these types.

Compatibility Considerations

Using the Intel IPP library for `uint16` images, poses no compatibility issues. The same is not true, however, for the `double` data type.

If an input image contains NaN values and a filtering kernel contains zero values, the `imfilter` function now gives different results when the Intel IPP library is enabled versus when it is disabled. If you want to preserve the behavior of previous releases, use `iptsetpref('UseIPPL', false)` to disable the Intel IPP library.

Improved Speed for Calculating N-D Euclidean Distance Transforms with the `bwdist` Function

A new algorithm improves the speed and reduces the memory footprint for the `bwdist` function.

Compatibility Considerations

In previous releases, the `bwdist` function used different algorithms for computing the Euclidean distance transform and the associated label matrix. If you need the same results produced by the previous implementation, use the function `bwdist_old`.

Modified Behavior for the `regionprops` ConvexHull Property

The 'ConvexHull' property of `regionprops` depends on the MATLAB `convhull` function. Due to changes in `convhull`, the results returned by 'ConvexHull' will now be slightly different than in previous releases.

Compatibility Considerations

The order of the vertices returned by the 'ConvexHull' property of `regionprops` may differ from that returned in releases before R2009b. Also, the returned hull may contain additional collinear points that were omitted in previous releases.

Efficient Display and Navigation of Very Large NITF-File Images in `imtool`

The `rsetwrite` function allows you to create multi-resolution image pyramids (R-Sets) that you can open in `imtool`. In previous releases, `rsetwrite` worked only with TIFF files. Now it accepts NITF files, as well, as long as they are Version 2.0 or greater, contain an uncompressed image, have integer data (no floating point data), and have three or fewer image bands. Finally, if a NITF file has more than one band of data, the data must be unsigned.

Performance Improvements

The performance of several existing toolbox functions has been improved in this release. In some cases, other toolbox functions call these functions and therefore will also benefit from these speed improvements.

Faster Functions

- `bwdist`
- `imcomplement`
- `imdilate`
- `imerode`
- `imfilter`
- `improfile`
- `imrotate`

Multithreaded Functions

- `applylut`
- `bwpack`
- `bwunpack`
- `imdilate`
- `imerode`
- `imreconstruct`

R2009a

Version: 6.3

New Features

Bug Fixes

Compatibility Considerations

Faster, Less Memory-Intensive Workflow for Labeling Regions and Measuring Their Properties

The `bwconncomp` function computes connected components for binary images. It uses significantly less memory and is sometimes faster than `bwlabel` and `bwlabeln`.

To extract features from a binary image using `regionprops` with default connectivity, just pass `BW` directly into `regionprops` using the command `regionprops(BW)`. To compute a label matrix having more memory-efficient data type, such as `uint8` versus `double`, use the `labelmatrix` function on the output of `bwconncomp`.

Multithreaded Implementation of `imfilter` Function

The `imfilter` function is now multithreaded.

Efficient Display and Navigation of Very Large Images in `imtool`

The new `rsetwrite` function allows you to create a multi-resolution image pyramid (R-Set) from a large TIFF image file. In previous releases, large images would not open in `imtool`, or they did open, but navigation was slow. You can now open your R-Set with `imtool` and explore it as you would a standard image.

New Dialog Box for Setting Toolbox Preferences

A new preferences dialog box allows customization of Image Processing Toolbox preferences. You can access the dialog box via the **File** menu in the MATLAB desktop, the **File** menu in the Image Tool (`imtool`), or directly from the command line by typing `iptprefs`.

A new preference has been added that allows you to specify whether the Overview tool opens automatically when you launch the Image Tool.

Compatibility Considerations

In previous releases, the Overview tool opened automatically with `imtool`. The new default behavior is for the Overview tool to no longer open automatically. If you would like to revert to the previous behavior you can set this preference via the Image Processing Preferences dialog box (`iptprefs`).

In previous releases, if you changed the preferences with the `iptsetpref` command, these changes would revert to the default setting when you finished a MATLAB session. Now, if you change preferences, these changes will remain intact from one MATLAB session to the next.

New `imcolormaptool` Function That Opens Choose Colormap Tool

The new function `imcolormaptool` opens the Choose Colormap tool. The Choose Colormap tool allows you to interactively change the colormap of a displayed image. You can also access the tool from the **Tools** menu of the Image Tool, as in previous releases.

End Point and Branch Point Detection Now Possible

`bwmorph` now detects end points and branch points in binary images.

nitfread Now Allows Image Subregion Selection

The `nitfread` function now includes a `PixelRegion` parameter that returns a subimage as specified by row and column vectors.

Compatibility Considerations

From R2007a to R2008b, the `nitfread` function returned `uint8` data for images with 1-bit data. Now `nitfread` returns `logical` data for images with 1-bit data. If you want this function to behave as it did in the past, enter the following:

```
imdata = uint8(nitfread(filename));
```

Support for Intel IPP on Mac

In previous releases, the Image Processing Toolbox leveraged the Intel Integrated Performance Primitives (Intel IPP) Library on 32- and 64-bit Linux® and Windows platforms. Now Intel IPP-use has been extended to the Mac.

getColor, getLabelVisible, and setLabelVisible Methods Added to imdistline

`imdistline` now includes a `getColor` method that returns the color used to draw a specific ROI object. Also, the new `getLabelVisible` and `setLabelVisible` methods make it possible to control the visibility of the Distance tool text label.

Five Functions Moved to MATLAB

The following five functions moved from the Image Processing Toolbox to MATLAB: `cmpermute`, `cmunique`, `dither`, `imapprox`, and `rgb2ind`. The behavior of some of the functions has changed slightly, as described in the compatibility considerations listed below.

Compatibility Considerations

- Functions `dither` and `imapprox`, when called without output arguments, no longer display their output as an image via a call to `imshow`. Now, if you want to display the resulting image, assign the output to one or more variables and call `imshow`. For example, try the following:

```
[Y,newmap] = imapprox(X,map,n)
imshow(Y,newmap)
```

- Function `rgb2ind` errors when called with the syntax `rgb2ind(RGB)`. You must specify the number of colors, tolerance, or colormap. For example, you can use the syntax `rgb2ind(RGB, 128)`, where 128 represents the number of colors.
- Function `imapprox` errors when called with the syntax `imapprox(x, map)`. As with `rgb2ind`, you must specify additional parameters.

Fan-Beam Functions Updated

Compatibility Considerations

Due to a bug fix, the fan-beam functions (`fanbeam`, `ifanbeam`, `fan2para`, `para2fan`) now return different answers than in previous releases.

R2008b

Version: 6.2

New Features

Bug Fixes

Compatibility Considerations

Performance Improvements

The performance of several existing toolbox functions has been improved in this release, including:

- Binary erosion and dilation (`imdilate`, `imerode`, `bwhitmiss`, and `rangefilt`)
- `graycomatrix`
- Image arithmetic and filtering now leverage the IPP Library on 32- and 64-bit Windows and Linux platforms.

New `cornermetric` Function Detects Corners

New `cornermetric` function detects corners.

Now Support Absolute Colorimetric Rendering Intent for GrayTRC and MatTRC

New additions to the `makecform` syntax include rendering intents for the Matrix/Tone Reproduction Curve (MatTRC) model and the single-channel Tone Reproduction Curve (GrayTRC) model.

New `createMask` Method Creates Mask for Any ROI

Use the new `createMask` method of the `imroi` base class to return a mask, or binary image, that is the same size as the input image with 1s inside the ROI object and 0s outside. The new method is available in the following classes: `impoint`, `imline`, `imrect`, `imellipse`, `impoly`, and `imfreehand`.

Interactive Tools Refresh when Target Image Changes

The following modular interactive tools now update automatically if you modify the target image: Adjust Contrast, Pixel Region, Pixel Information, Overview, Display Range, and Image Information.

The `imscrollpanel` 'PreserveView' Parameter Now Works for Images of All Sizes

The `replaceImage` function in the `imscrollpanel` API has been modified. You can now use the 'PreserveView' parameter even in cases where your replacement image is not the same size as your original image. The new image will appear with the center of view in the same relative position as in the original image.

Compatibility Considerations

In previous releases, the default for different size images was for the new image to appear centered and at 100% magnification.

Distance Tool and Cropping Tool Now Modes in `imtool`

The Distance tool and Cropping tool have been modified. Now to use the Distance tool, you click one end of the distance to be measured, drag, and release to complete the measurement. With the new

version of the Cropping tool, you may click and drag to define the cropping region as many times as you want. If you define one region and then decide to crop a different region instead, simply click and drag the mouse again to define the new region.

Compatibility Considerations

In previous releases, the Distance tool appeared as a horizontal bar of set length. You could drag the ends of the tool to change size and orientation.

In previous releases, if you defined one cropping region, you could move this box or change the size, but you couldn't start with a new box unless you canceled the tool, clicked on the "Crop Image" toolbar button, and then defined the new region.

In imtool Opening Adjust Contrast Tool No Longer Selects Window/Level Tool

When you open the Adjust Contrast tool, the Window/Level tool is no longer turned on automatically. To operate this feature, simply select the Window/Level tool icon from the Image Tool toolbar. (To identify the icon, note that if you move the cursor over the Window/Level tool icon, the words "Adjust contrast/brightness via mouse motion" appear.) Or, you can select "Tools" from the Image Tool menu and then click on "Window/Level."

Compatibility Considerations

In previous releases, when you opened the Adjust Contrast tool, the Window/Level tool automatically turned on at the same time. Note that the Window/Level tool still turns on when you call `imcontrast` from the command line.

immovie Command No Longer Shows Preview

`immovie` no longer opens a figure window to display the movie as it is being created. You can display and explore the output of `immovie` using `implay`.

Compatibility Considerations

If you want to use `movie` to visualize the output but don't know how to set up the figure appropriately, call `imshow` on one of the movie frames first before calling `movie`.

Replace Calls to ipttable Function with MATLABuitable Function

The `ipttable` function is being deprecated and will be removed in a future release.

Compatibility Considerations

If you used the `ipttable` function to display tabular data, you should replace use of `ipttable` with the MATLAB function `uitable`.

If you used the cell array syntax of `ipttable`, make the following changes to your code to achieve a similar effect using `uitable`. For more information about using `uitable`, see the `uitable` function reference page.

R2008a Code	R2008b Code
<pre>table = ipttable(parent,cell_array_data);</pre>	<pre>table = uitable(parent,'Data',cell_array_data);</pre>

If you used the struct syntax of `ipttable`, make the following changes to your code to achieve a similar effect with `uitable`.

R2008a Code	R2008b Code
<pre>table = ipttable(parent,struct_data);</pre>	<pre>field_names = fieldnames(struct_data); values = struct2cell(struct_data); for idx = 1:numel(values) val = values{idx}; if ~ischar(val) size(val,1) > 1 values{idx} = evalc('disp(values{idx})'); end end table = uitable(parent,'Data', [field_names values]);</pre>

Code written in previous releases that depends on `ipttable` will begin to warn and eventually error in later releases.

imcontour Second Output Argument Changed

The second output argument of `imcontour` is now a handle to an `hggroup` object instead of an array of handles to patch objects.

Compatibility Considerations

If you need to access handles of individual patch objects, use the following code to work around the change.

```
[c,handleToHGGroup] = imcontour(...);
arrayOfHandlesToPatchObjects = get(handleToHGGroup,'Child');
```

impixelinfo Tool Disappears when Image Changes

If you use `imshow` to display an image, open the `impixelinfo` tool, and use `imshow` to open a new image, the `impixelinfo` tool will disappear along with the first image.

Compatibility Considerations

In previous versions, if you entered the following code:

```
imshow pout.tif
impixelinfo
imshow peppers.png
```

the `impixelinfo` tool would update to reflect changes to the image. Now you must call the `impixelinfo` tool again after opening the second image.

Some Code Moved into Different Directories

- Colorspace functionality moved into the new `toolbox/images/colorspaces` directory.
- Medical file formats moved into the `toolbox/images/iptformats` directory with other file formats, and the `toolbox/images/medformats` directory was removed.

Functions and Demos Being Removed

Function or Demo Name	What Happens When You Use Function or Demo?	Use This Instead	Compatibility Considerations
<code>pixval</code>	Errors	Use <code>impixelinfo</code> for pixel reporting and <code>imdistline</code> for measuring distance.	Replace all existing instances of <code>pixval</code> with <code>impixelinfo</code> or <code>imdistline</code> .
<code>dctdemo</code>	Errors	NA	None
<code>edgedemo</code>	Errors	NA	None
<code>firdemo</code>	Errors	NA	None
<code>landsatdemo</code>	Errors	NA	None
<code>nrfiltdemo</code>	Errors	NA	None
<code>qtdemo</code>	Errors	NA	None
<code>roidemo</code>	Errors	NA	None

R2008a

Version: 6.1

New Features

Bug Fixes

Compatibility Considerations

Create High Dynamic Range (HDR) Images and Write Them to Files

Create a high dynamic range image from a group of low dynamic range images using the new `makehdr` function. The low dynamic range images must be spatially registered. You can write the HDR image to a file using the `hdrwrite` function. These functions complement the `hdrread` and `tonemap` functions introduced in R2007b.

Measure Properties of Regions in Grayscale Images

The `regionprops` function now accepts grayscale images as an input parameter, returning measurements based on the values of pixels in specified regions. Using `regionprops`, you can obtain measurements of regions in the image such as the maximum, minimum, and mean intensities in the region, and the weighted centroid.

Display Very Large Images by Subsampling

You can now display very large images from TIFF files by using the `imshow` function's new 'Reduce' parameter. When you specify this parameter, `imshow` displays a subsampled version of the image. `imshow` determines the subsampling factor by considering the size of the image and the reduction required to fit the image on your screen. The 'Reduce' parameter makes it possible to view very large images in their entirety that could not previously be displayed. Note, however, that the image subsampling that is performed reduces that amount of image data displayed.

Enhancements to ROI Tools

The toolbox includes several functions that enable the definition of regions of interest of various shapes: `impoint`, `imline`, `impoly`, `imrect`, and `imfreehand`. These ROI tools have several enhancements:

ROI Tools Reimplemented as MATLAB Classes

The ROI tools have been reimplemented as MATLAB classes. This change does not affect how the ROI tools function; they function identically to their previous implementation. The documentation uses the MATLAB functional syntax descriptions rather than the dot notation. That is, the documentation shows how to call the class methods specifying a handle to the object as the first argument, `method(h, ...)`. Note, however, that you can still use the dot notation when calling the methods, `obj.method(...)`. In addition, the `iptgetapi` function now returns an object of the new class which means that code similar to the following will continue to work:

```
api = iptgetapi(h)
api.method()
```

Compatibility Considerations

The class of the data returned by the ROI tools is now a handle to an ROI class, such as `imline` or `impoly`. In addition, several undocumented methods supported by the ROI tools have been removed: `getContextMenu`, `setContextMenu`, `getDrawAPI`, `addCallback`, and `removeCallback`.

ROI Tools Support New wait and resume Methods

The ROI tools now support `wait` and `resume` methods so that they can be used in scripts. By using the `wait` method, you can enable users of your script to make the initial placement of the ROI, adjust

the ROI and accept it, and then use the position in the script. For example, using the `wait` method with an ROI tool, you could write a script that creates a mask.

The `resume` method is a programmatic way to return control to the command line. When called after `wait`, `resume` causes `wait` to return the accepted position of the ROI.

Interactively Add New Vertices to ROI Polygons

You can now add vertices interactively to polygonal ROIs that you define using the `impoly` function. To create the new vertex, position the pointer over an edge of the polygon and press the A key. The pointer changes shape. Click the mouse to add a new vertex. The `roifill` and `roipoly` functions, which use `impoly` to implement ROIs, also support this new capability.

Enhancements to Color Functions

The following color functions have been enhanced.

makecform Supports Converting Between sRGB and CMYK

The `makecform` function now supports two new color space conversion types for converting between sRGB and CMYK: `'srgb2cmyk'` and `'cmyk2srgb'`.

iccwrite Creates Smaller ICC Profiles

The `iccwrite` function now uses certain optimizations to reduce the size of the International Color Consortium (ICC) color profiles that it creates. `iccwrite` uses aliasing to avoid writing tag data multiple times when it is included in more than one profile table.

cp2tform Function Supports New Transformations

The `cp2tform` function supports two new transformation types: `'similarity'` and `'nonreflective similarity'`.

Compatibility Considerations

The `'linear conformal'` transformation type supported by the `cp2tform` function has been renamed to `'nonreflective similarity'`.

hough Function Uses Specified RhoResolution Values

The `hough` function now uses the value you specify for the `'RhoResolution'` parameter. In previous releases, the function did not use the value specified.

Compatibility Considerations

The Hough matrix, `H`, and the `Rho` outputs returned by the `hough` function have different results than those obtained from the same function in previous releases.

Enhancements to Interactive Tools

The following modular interactive tools have been enhanced.

- Adjust Contrast tool (`imcontrast`) — The **Adjust Data** button in the Adjust Contrast tool is disabled until you make a change to image contrast.
- Pixel Region tool (`impixelregion`) — To improve the visibility of the image pixels being examined, the Pixel Region tool stops including grid lines in the display at low magnifications.

New and Updated Demos

The toolbox includes the following new and changed demos.

- *Batch Processing Image Files in Parallel* is an existing demo that has been updated, and simplified, through use of the `parfor` function.
- *Detecting Cars in a Video of Traffic* is a new demo that shows how to use the toolbox to visualize and analyze videos or image sequences.
- *Measuring Regions in Grayscale Images* is a new demo that shows how to use the `regionprops` function with grayscale images.

Enhancements to Other Functions

This release includes changes to the following functions.

Function	Description of Enhancement
<code>imageinfo</code>	Accepts files of several additional file formats as an input argument, including NITF, Interfile, and Analyze file formats.
<code>imshow</code>	Supports a new <code>colormap</code> parameter for specifying a colormap for grayscale images.
<code>imtool</code>	Supports a new <code>colormap</code> parameter for specifying a colormap for grayscale images.
<code>trueSize</code>	Preserves the border preference setting of the figure when adjusting the image display size.

R2007b

Version: 6.0

New Features

Bug Fixes

Compatibility Considerations


New Interactive Image Sequence and Video Viewer

The toolbox now supports a new interactive image sequence viewer, called the Movie Player (`implay`). Using the Movie Player you can:

- Play a MATLAB movie, AVI file, or multidimensional array.
- Step through a movie or sequence of images, frame-by-frame, or jump to the beginning or end of the sequence.
- Examine a frame using the Pixel Region tool or export the frame to the Image Tool.

Image Tool Includes Cropping, Enhanced Contrast Adjustment, and Saving of Modified Images

The Image Tool (`imtool`) supports several enhancements:

- You can now modify the image data after performing a contrast adjustment operation. Previously, contrast adjustment only affected the display of the image, not the actual image data. To modify image data, click **Adjust Data** in the Adjust Contrast tool.
- You can now interactively crop an image displayed in the Image Tool using the Crop Image button  in the toolbar (or select **Crop Image** from the Tools menu).
- You can now save the image displayed in the image tool in any of several common image file formats. Select **Save As** from the Image Tool File menu.

New Function for Converting Bayer Pattern Encoded Images to RGB

The toolbox now supports a function, `demosaic`, that can convert a Bayer pattern encoded image into an RGB image.

A Bayer filter mosaic, or color filter array, refers to the arrangement of color filters that let each sensor in a single-sensor digital camera record only red, green, or blue data. The patterns emphasize the number of green sensors to mimic the human eye's greater sensitivity to green light. The `demosaic` function uses interpolation to convert the two-dimensional Bayer-encoded image into a truecolor image, in the RGB color space.

New Function for Creating a Multiresolution Gaussian Pyramid

The toolbox now supports a function, `impyramid`, that you can use to create a multiresolution Gaussian pyramid. If you specify the 'reduce' parameter, `impyramid` returns a low-pass filtered version of the image, half the size of the original image. If you specify the 'expand' parameter, `impyramid` returns a filtered image twice the size of the original image. `impyramid` uses convolution with a Gaussian filter kernel to produce the images.

Enhanced ROI Definition Behavior for `imcrop`, `roifill`, and `roipoly`

The `imcrop`, `roifill`, and `roipoly` functions now let you define an ROI and then adjust the size and position of the ROI interactively using the mouse. In previous releases, these functions supported the interactive definition of ROIs, but only gave you one chance at the definition. Now, when you are satisfied with the size and shape of the ROI, double-click to perform the cropping, filling, or mask creation operation.

Compatibility Considerations

In previous releases, when defining a polygonal ROI using `roipoly`, pressing **Backspace** deleted the most recent vertex you had defined in the polygon. With this release, pressing **Backspace** deletes the entire polygon. To delete an individual vertex, move the pointer over the vertex, right-click to view the vertex context menu, and then choose **Delete Vertex**.

New Modular Interactive GUI-building Tools

The set of modular interactive tools now includes functions to display a file chooser dialog box and write data to a file (`imsave`). The toolbox also includes a function (`inputfile`) that displays the file chooser dialog box and returns the user's selections.

New Programmable ROI Tools

The set of programmable ROI creation functions provided by the toolbox now includes three additional shapes:

- Polygons (`impoly`)
- Ellipses (`imellipse`)
- Freehand shapes (`imfreehand`)

The toolbox already includes ROI creation functions to create points, lines, and rectangles.

Each of the ROI creation functions supports an API that you can use to control aspects of its behavior and appearance. For example, you can use API functions to specify the position of the ROI or retrieve the coordinates of its current position.

Support for Reading NITF and HDR Images

- Read metadata from a National Imagery Transmission Format (NITF) file using `nitfinfo`.
- Read an image from a NITF file using `nitfread`.
- Read high dynamic range (HDR) images using `hdrread`.
- Convert high dynamic range images into a format that can be displayed using the `tonemap` function.

Enhanced Performance

- Enhanced performance for thinning and skeletonization using `bwmorph`.
- Enhanced performance for filtering RGB images using `imfilter`.

DICOM Dictionary Upgrade

The default DICOM dictionary has been upgraded to the 2007 version released by NEMA. A text version of this dictionary is included in the product, `dicom-dict.txt`. This upgrade fixes a problem with the earlier version of the dictionary which contained two instances of the same tag, which caused warnings.

Compatibility Considerations

If your DICOM code depends on hard-coded old attribute names, you may see failures. In addition, some DICOM files may no longer parse. Customers who require attribute settings from the 2005 version can use the `dicomdict` function to access the old data dictionary, which we are shipping in R2007b. That is, `dicom-dict.txt` will have 2007 values and `dicom-dict-2005.txt` is the version of `dicom-dict.txt` found in R2006a and R2007a.

Changes to Other Functions

This release includes changes to the following functions.

Function	Description of Change
<code>imshow</code>	Is not supported when MATLAB is started with the <code>-nojvm</code> option.
<code>imhist</code>	Can now be embedded in custom GUIs.
<code>fanbeam</code> , <code>ifanbeam</code> , <code>fan2para</code> , <code>para2fan</code>	The fan-beam functions now return different answers than in previous releases due to a bug fix.
<code>imadjdemo</code>	This demo has been deleted from the toolbox.

R2007a

Version: 5.4

New Features

Bug Fixes

Compatibility Considerations

Enhancements to `imresize` Function

`imresize` now runs faster, uses less memory, supports new interpolation methods, and supports new options for specifying output size.

Compatibility Considerations

The `imresize` function has been completely rewritten with new algorithms, new options, and new syntaxes. If you need the results produced by the version of `imresize` in previous releases, use the `imresize_old` function.

`applycform` Supports Tetrahedral Interpolation

The `applycform` function now uses tetrahedral interpolation for profiles containing multidimensional lookup tables, and returns more accurate results.

Compatibility Considerations

The results returned by `applycform` are more accurate but they are different than results returned in previous releases, for profiles containing multidimensional lookup tables.

Control Point Selection Tool Enhancements

The Control Point Selection Tool has enhanced visual appearance and usability. For example, points are now numbered for easier identification of matched pairs.

In addition, the tool now supports a 'wait' option which enables `cpselect` to be used in scripts. When you specify this option, `cpselect` blocks the MATLAB command line until point selection is completed. For information about using the Control Point Selection Tool, see Image Registration and the reference page for the `cpselect` function.

Compatibility Considerations

The Control Point Selection Tool no longer includes the **Redo** or **Undo** options on the Edit menu.

Enhancements to `impoint`, `imline`, and `imrect` Functions

The `impoint`, `imline`, and `imrect` function now support an interactive placement capability. Using the mouse, you can specify the initial position of the point, line, or rectangle. In addition, the `imrect` function now supports interactive resizing using the mouse. See the reference pages for these functions for more information and examples.

Enhancements to `montage` Function

The `montage` function now supports parameters that control the arrangement and appearance of the images displayed. See the `montage` reference page for these functions for more information and examples

makecform Uses 'icc' Whitepoint for L*a*b*/sRGB Conversions

The `makecform` function now only uses the white point type 'icc' for color space conversions from $L^*a^*b^*$ to *srgb* (type = 'lab2srgb') and from *srgb* to $L^*a^*b^*$ (type = 'srgb2lab').

Compatibility Considerations

In previous releases, you could specify other white point values for these conversions, using the optional 'Whitepoint' parameter. This syntax now issues a warning when any other white point besides 'icc' is specified.

normxcorr2 Might Return Different Results

The `normxcorr2` function now returns values in the range [-1, 1] for all inputs. In previous releases, `normxcorr2` returned values outside this range for certain inputs.

watershed Uses New Algorithm

The watershed transform algorithm used by the `watershed` function has changed. The previous algorithm occasionally produced labeled watershed basins that were not contiguous..

Compatibility Considerations

If you need to obtain the same results as the previous algorithm, use the function `watershed_old`.

Changes to Other Functions

This release includes changes to the following functions.

Function	Description of Change
<code>imshow</code>	New 'border' parameter, to control whether <code>imshow</code> includes a border around the image displayed, and 'parent' parameter, to specify the axes in which to display the image.
<code>imshow_scrollpanel</code>	New 'replaceImage' parameter lets you replace the image displayed in the scroll panel with a new image.
<code>iradon</code>	New 'none' value for the <code>filter</code> parameter returns an unfiltered backprojection; also supports new interpolation types.
<code>iptsetpref</code>	New UseIPPL preference.

R2006b

Version: 5.3

New Features

Bug Fixes

Compatibility Considerations

Enhancements to DICOM Capabilities

This release includes the following new features and enhancements to the DICOM capabilities of the Image Processing Toolbox:

- The toolbox includes a new function, `dicomlookup`, that provides a way to find the name of an attribute in a DICOM data dictionary by specifying its group and element tags, or find the group and element tags for an attribute by specifying its name.
- The performance of the `dicominfo` function has been significantly improved

New Symmetric Option with `graycomatrix` Function

The `graycomatrix` function now supports a new option: `'symmetric'`. With this option, you can create a gray-level co-occurrence matrix (GLCM) that is symmetric about its diagonal. This is consistent with the GLCM definition given by Haralick in his 1973 article. For more information, see `graycomatrix`.

Enhancements to ICC Color Capabilities

The toolbox includes the following enhancements to the ICC color capabilities:

- The `applycform` function can now transform colors using profiles that contain parametric curve types.
- The `iccread` function now supports named colors in ICC profiles.

Compatibility Considerations

The `whitepoint` function, when used with the `'d50'` argument, returns different results in R2006b than it did in R2006a. The previously returned XYZ color values were incorrect according to the current interpretation of standards. If your algorithm depended on the old values, you might see subtly different results.

Enhancements to the `imdistline` Function

This release includes the following enhancements to the `imdistline` function:

- The `imdistline` function now uses a different cursor shape at its endpoints to highlight that these endpoints can be grabbed to change the length or direction of the line. The function uses a hand cursor over endpoints and a fleur cursor over the body of the line.
- The `imdistline` function reference page now includes an example that shows how to use the `XData` and `YData` properties of the associated image to express distance in non-pixel units.

Compatibility Considerations

The Distance Tool's `getAngleFromHorizontal` method now returns a value between 0 and 180 degrees. Previously, this function incorrectly returned a value between 0 and 90. For an explanation of how `getAngleFromHorizontal` calculates this angle, see the `imdistline` function.

setColor Method Accepts Predefined Color Strings

The `setColor` method of the `imshow`, `imline`, `impoint`, and `imrect` functions accepts an RGB triplet or the short- or long-name version of the MATLAB predefined color names.

R2006a

Version: 5.2

New Features

Bug Fixes

Compatibility Considerations

Enhanced ICC Profile Capabilities

The `iccread` and `iccwrite` functions have been updated to support recent changes to the ICC specification.

In addition, `iccread` can now read and process the following additional profile types:

- DeviceLink profiles — Provide transformation from one device space to another.
- ColorSpace profiles — Provide transformation between a non-device color space and the profile connection space (PCS).
- Abstract profiles — Enable color transformations to be defined that provide specific color effects.
- Grayscale profiles — Specify the relationship between device values and the PCS for specific colors.

In addition, `iccread` can now read parametric curve types.

New Pointer Management Functions

The toolbox includes three new utility functions, `iptPointerManager`, `iptGetPointerBehavior`, and `iptSetPointerBehavior`, that you can use to manage changes to the pointer in GUIs. For example, you can use the pointer management functions to change the appearance of the pointer when it moves over objects in a figure. These functions can be useful when building GUIs with the toolbox modular GUI tools.

New Constraint Creation Function

The toolbox includes a new utility function, `makeConstrainToRectFcn`, that you can use to specify drag constraints for the `imdistsline`, `imline`, `impoint`, and `imrect` functions. You specify the constraints as arguments to the `makeConstrainToRectFcn` and this function returns a handle to a constraint function. To use this constraint with an object, set the value of the `setConstraintFcn` API for the object to this function handle.

Functions `cp2tform`, `tforminv`, `imtransform`

When using the `cp2tform`, `tforminv`, or `imtransform` functions with the transform type `'piecewise linear'` you might get different answers from previous versions due to a bug fix. If you have a transformation structure (TFORM) saved from an older version, you may want to regenerate it from control points to get improved performance.

Compatibility Considerations

If you have a transformation structure (TFORM) saved from an older version, you may want to regenerate it from control points to get improved performance.

IPPL Not Used on 64-Bit Systems

Certain functions in the Image Processing Toolbox, such as the image arithmetic functions, use the Intel Performance Primitives Library (IPPL), if it's available. (See `ippl` for more information.) Note that these functions do not use the IPPL on 64-bit systems.

R14SP3

Version: 5.1

New Features

Bug Fixes

Compatibility Considerations

Support for Two New Medical Image File Formats

The toolbox now includes functions for reading metadata and image data from two additional medical image file formats. Analyze 7.5 and Interfile. For more information, see “Mayo Analyze 7.5 Files” and “Interfile Files”.

New Point, Rectangle, and Line Functions

The toolbox includes three functions, `impoint`, `imline`, and `imrect`, that you can use to create draggable points, lines, and rectangles in a figure window. These functions can be used as building blocks for other GUI tools.

Image Tool Enhancements and Improvements

New Distance Tool

The Image Tool now includes a new Distance tool that you can use to determine the distance between any two points in an image. This tool is also available in the toolbox's suite of modular interactive GUI tools. Using the `imdistanline` function you can add the Distance tool to GUIs of your own creation. For more information, see Measuring Features in an Image

Adjust Contrast Tool Enhancements and Improvements

The Adjust Contrast tool has been redesigned to provide better usability. For examples, the Adjust Contrast tool Window/Level capability is now a separate mode with its own activation button.

New Utility Functions for Use with Profile-Based Color Space Conversion Functions

The toolbox has two new utility functions, `iccroot` and `iccfnd`, for use with the profile-based color conversion functions. For more information, see Performing Profile-based Color Space Conversions.

New Documentation on Processing Image Sequences

The Image Processing Toolbox User's Guide includes a new section, Working with Image Sequences, that describes which toolbox functions can be used with sequences of image, also known as image stacks

Control Point Selection Tool Now Works on Macintosh Systems

The Control Point Selection Tool now works on Macintosh systems.

Obsolete and Deleted Functions

Compatibility Considerations

The following table lists toolbox functions that have been made obsolete or removed in this version.

Function	Enhancement
<code>impositionrect</code>	This function is obsolete. Use <code>imrect</code> to perform the same tasks.
<code>pixval</code>	This function is obsolete. It now issues a warning when used. Use <code>impixelinfo</code> for pixel reporting and use <code>imdistline</code> for measuring distance

Image Tool is Not Compilable

Compatibility Considerations

The `imtool` function is not compilable with the MATLAB Compiler.

R14SP2

Version: 5.0.2

New Features

Major Bug Fixes

This release contains the following bug fixes.

Major Revisions to Fan-Beam Functions

This release includes numerous updates and improvements to the fan-beam functions: `fanbeam`, `ifanbeam`, `fan2para`, and `para2fan`. The fixes include improved calculations, improved documentation, and examples.

For example, `fanbeam` now returns the correct sensor locations when the geometry is 'line'. The `ifanbeam` and `fan2para` now consistently use the correct default value for the 'FanSensorSpacing' parameter. If you tried the fan-beam functions in a previous release, you might try them again to take advantage of these improvements.

In addition to the functional changes, many improvements to the documentation of the fan-beam functions have been made.

`fanbeam` help now includes

- An example that shows how to extract projection data at a specific rotation angle from the fan-beam data returned
- An explanation of how `fanbeam` calculates the number of rows and columns in `F`, the fan-beam data returned
- The default value for the 'FanSensorSpacing' parameter for both 'line' and 'arc' geometries
- Guidelines for setting the value of the `D` parameter

The help for the `ifanbeam` function now includes an example that shows how to use the 'minimal' coverage parameter.

Compatibility Considerations

Results computed with earlier versions of the fan-beam functions cannot be used with the new versions of these functions.

Changes to the DICOM Functions

The following fixes have been made to the `dicomread` and `dicomwrite` functions.

Function	Bug Fixes
<code>dicomread</code>	No longer errors when reading files that contain extraneous pixel data; instead, <code>dicomread</code> issues a warning message. However, if the file does not contain enough pixel data, <code>dicomread</code> issues an error.

Function	Bug Fixes
dicomwrite	<ul style="list-style-type: none"> • No longer is case sensitive when parsing input parameters. For example, you can specify either 'CreateMode' or 'createmode'. • Preserves the full precision of data converted to decimal string metadata. Previously, dicomwrite limited precision to six digits. • No longer errors when writing files with metadata values that must be stored as a decimal string or integer string. Now, when writing private data attributes (attributes that are not listed in the DICOM data dictionary), dicomwrite assigns the attributes the type UN (for unknown) and writes the data to the file as a byte-for-byte copy of its in-memory representation. Because dicomwrite writes the file with explicit value representation (VR), the file might have a different VR value, but the data will be the same. • Includes the TriggerTime field for additional values of ScanOptions, including 'CT'. Previously, dicomwrite only included the TriggerTime attribute if the ScanOptions field indicated a gated heart MR. • No longer issues an Unsupported SOP class error message if, when 'create' mode is specified, semantic verification is not available for an information object. Instead, dicomwrite issues a more helpful message indicating that it might be able to write the data if the mode was 'copy', rather than 'create'. In 'copy' mode, dicomwrite only performs syntactic checking, not semantic verification. Consequently, dicomwrite can write many more types of DICOM files in 'copy' mode than it can in 'create' mode. See the dicomwrite reference page for important information about data integrity.

Changes to Image Tool and Modular Interactive Tools

The following fixes have been made to the Image Tool and other modular interactive tools:

- The Image Tool now always makes the **Open** and **Import from Workspace** options available on its **File** menu. Previously, the Image Tool disabled these options if the tool contained an image. If the Image Tool contains an image, the newly imported image is displayed in a new Image Tool using the default preferences.
- The Image Tool zoom buttons can now be used on an image that has superimposed vector data.
- The Image Tool toolbar buttons no longer create multiple versions of the modular interactive tools when clicked rapidly in quick succession.
- The Image Information tool now displays correctly on Linux systems. Previously, it displayed as a blank window.
- The Overview tool can now be resized from any corner. Previously, resizing the tool using a corner other than the lower left caused the image to become progressively smaller until it disappeared.
- The Overview tool zoom buttons now provide an affordance that informs users when they cannot use these buttons to zoom in or out on the image displayed in the associated scroll panel.
- The Pixel Region tool now displays floating-point values correctly. Previously, the pixel value text strings displayed spilled over into adjacent pixels for some floating-point images.
- The Pixel Region tool now works correctly with images displayed in subplots.
- The Pixel Region tool no longer causes the target image to become tiny and move to a different position in the figure.

Changes to the imshow Function

- The `imshow` function no longer overwrites nondefault axes in a figure.
- The `imshow` function ignores any initial magnification value you specify when used to display an image in a figure that is docked (the figure's `WindowState` property is set to `'docked'`). In these cases, `imshow` displays the image at the largest magnification that fits the window (`'fit'` magnification) and issues a warning.

Fixes to Other Functions

The following tables lists fixes that have been done to other toolbox functions.

Function	Enhancement
<code>applycform</code>	Now correctly handles profiles that contain a <code>gamut</code> tag.
<code>cpcorr</code>	Now is more numerically robust. For this release, the subfunction <code>findpeak</code> , which <code>cpcorr</code> calls, has been improved and is now a private function, rather than a subfunction.
<code>imhist</code>	No longer causes a docked figure window to become undocked.
<code>imrotate</code>	Now correctly rotates N-dimensional arrays, where N is greater than 3. In previous releases, <code>imrotate</code> would accept N-D arrays but only return a 3-D array.
<code>normxcorr2</code>	Now always returns real values. In previous releases, due to round-off error, some sets of input data caused the <code>normxcorr2</code> function to return a complex valued matrix of correlation coefficients.
<code>pixval</code>	Now works correctly with binary images.
<code>rgb2ind</code>	Now returns a correct output image when called with the syntax <code>rgb2ind(rgb,n,'nodither')</code> where n is greater than 256.

Fixes to Image Processing Toolbox Deployment Issues

- Performance issues that occurred when deploying compiled image processing toolbox functions that call IPPL routines have been fixed.
- Running compiled versions of `imtool` and some of the other modular interactive tools no longer generates the following warning messages about classes not being cleared:

```
Warning: Objects of graphics.linkprop class exist - not clearing
this class or any of its super-classes.
Warning: An object instance still exists. Use the objectdirectory
command to see a count of existing instances.
```

New Directory Needed

The Image Processing Toolbox software now requires the following new directory on the MATLAB path:

```
toolbox\shared\imageslib
```