imwrite

Write image to graphics file

Syntax

```
imwrite(A,filename)
imwrite(A,map,filename)

imwrite( __ ,fmt)

imwrite( __ ,Name,Value)
```

example

example

example

Description

`imwrite(A,filename)` writes image data `A` to the file specified by `filename`, inferring the file format from the extension. `imwrite` creates the new file in your current folder. The bit depth of the output image depends on the data type of `A` and the file format. For most formats: example

- If `A` is of data type `uint8`, then `imwrite` outputs 8-bit values.
- If `A` is of data type `uint16` and the output file format supports 16-bit data (JPEG, PNG, and TIFF), then `imwrite` outputs 16-bit values. If the output file format does not support 16-bit data, then `imwrite` returns an error.
- If `A` is a grayscale or RGB color image of data type `double` or `single`, then `imwrite` assumes that the dynamic range is [0,1] and automatically scales the data by 255 before writing it to the file as 8-bit values. If the data in `A` is `single`, convert `A` to `double` before writing to a GIF or TIFF file.
- If `A` is of data type `logical`, then `imwrite` assumes that the data is a binary image and writes it to the file with a bit depth of 1, if the format allows it. BMP, PNG, or TIFF formats accept binary images as input arrays.

If `A` contains indexed image data, you should additionally specify the `map` input argument.

`imwrite(A,map,filename)` writes the indexed image in `A` and its associated colormap, `map`, to the file specified by `filename`. example

- If `A` is an indexed image of data type `double` or `single`, then `imwrite` converts the indices to zero-based indices by subtracting 1 from each element, and then writes the data as `uint8`. If the data in `A` is `single`, convert `A` to `double` before writing to a GIF or TIFF file.

`imwrite( __ ,fmt)` writes the image in the format specified by `fmt`, regardless of the file extension in `filename`. You can specify `fmt` after the input arguments in any of the previous syntaxes.

`imwrite( __ ,Name,Value)` specifies additional parameters for output GIF, HDF, JPEG, PBM, PGM, PNG, PPM, and TIFF files, using one or more name-value pair arguments. You can specify `Name,Value` after the input arguments in any of the previous syntaxes. example

Examples

collapse all

Write Grayscale Image to PNG

Write a 50-by-50 array of grayscale values to a PNG file in the current folder.

Open This Example

```
A = rand(50);
imwrite(A,'myGray.png')
```

Write Indexed Image Data to PNG

Write an indexed image array and its associated colormap to a PNG file.

Load sample image data from the file, `clown.mat`.

```
load clown.mat
```

The image array `X` and its associated colormap, `map`, are loaded into the MATLAB® workspace.

Write the data to a new PNG file.

```
imwrite(X,map,'myclown.png')
```

`imwrite` creates the file, `myclown.png`, in your current folder.

View the new file by opening it outside of MATLAB.

Write Indexed Image with MATLAB Colormap

Write image data to a new PNG file with the built-in MATLAB colormap, `copper`.

Load sample image data from the file `clown.mat`.

```
load clown.mat
```

The image array `X` and its associated colormap, `map`, are loaded into the MATLAB workspace. `map` is a matrix of 81 RGB vectors.

Define a copper-tone colormap with 81 RGB vectors. Then, write the image data to a PNG file using the new colormap.

```
newmap = copper(81);
imwrite(X,newmap,'copperclown.png');
```

`imwrite` creates the file, `copperclown.png`, in your current folder.

View the new file by opening it outside of MATLAB.

Write Truecolor Image to JPEG

Create and write truecolor image data to a JPEG file.

Create a 49-by-49-by-3 array of random RGB values.

```
A = rand(49,49);
A(:,:,2) = rand(49,49);
A(:,:,3) = rand(49,49);
```

Write the image data to a JPEG file, specifying the output format using the string, `'jpg'`. Add a comment to the file using the `'Comment'` name-value pair argument.

```
imwrite(A,'newImage.jpg','jpg','Comment','My JPEG file')
```

View information about the new file.

```
imfinfo('newImage.jpg')
```

```
ans =

           Filename: 'S:\newImage.jpg'
        FileModDate: '25-Jan-2013 16:18:41'
           FileSize: 2339
             Format: 'jpg'
      FormatVersion: ''
              Width: 49
             Height: 49
           BitDepth: 24
          ColorType: 'truecolor'
    FormatSignature: ''
    NumberOfSamples: 3
       CodingMethod: 'Huffman'
      CodingProcess: 'Sequential'
            Comment: {'My JPEG file'}
```

Write Multiple Images to TIFF File

Write multiple images to a single multipage TIFF file.

Open This Example

Create two sets of random image data, `im1` and `im2`.

```
im1 = rand(50,40,3);
im2 = rand(50,50,3);
```

Write the first image to a new TIFF file. Then, append the second image to the same file.

```
imwrite(im1,'myMultipageFile.tif')
imwrite(im2,'myMultipageFile.tif','WriteMode','append')
```

Write Animated GIF

Animate a series of plots and write the result to a GIF file.

Define the x-axis limits for the plot and specify the output file name.

```
x = 0:0.01:1;
figure
filename = 'testAnimated.gif';
```

Call `frame2im` to get image data from a single movie frame. Because three-dimensional data is not supported for GIF files, call `rgb2ind` to convert the RGB data in the image data, `im`, to an indexed image, `A`, with a colormap, `map`. Call `imwrite` with the name-value pair argument, `'WriteMode','append'`, to append multiple images to the first image.

```
for n = 1:0.5:5
y = x.^n;
plot(x,y)
drawnow
frame = getframe(1);
im = frame2im(frame);
[A,map] = rgb2ind(im,256);
        if n == 1;
                imwrite(A,map,filename,'gif','LoopCount',Inf,'DelayTime',1);
        else
                imwrite(A,map,filename,'gif','WriteMode','append','DelayTime',1);
        end
end
```

`imwrite` writes the GIF file to your current folder. Name-value pair `'LoopCount',Inf` causes the animation to continuously loop. `'DelayTime',1` specifies a 1-second delay between the display of each image in the animation.

Input Arguments

collapse all

A — Image data to write
matrix

Image data to write, specified as a full (nonsparse) matrix.

- For grayscale images, `A` can be `m`-by-`n`.

- For indexed images, `A` can be `m`-by-`n`. Specify the associated colormap in the `map` input argument.
- For truecolor images, `A` must be `m`-by-`n`-by-3. `imwrite` does not support writing RGB images to GIF files.

For TIFF files, `A` can be an `m`-by-`n`-by-4 array containing color data that uses the CMYK color space.

For multiframe GIF files, `A` can be an `m`-by-`n`-by-1-by-`p` array containing grayscale or indexed images, where `p` is the number of frames to write. RGB images are not supported in this case.

Data Types: `double` | `single` | `uint8` | `uint16` | `logical`

`filename` — Name of output file
string

Name of the output file including the file extension, specified as a string. For a list of the image types that `imwrite` can write, see the description for the fmt input argument.

Example: `'myFile.gif'`

Data Types: `char`

`map` — Colormap of indexed image
`m`-by-3 array

Colormap associated with indexed image data in `A`, specified as an `m`-by-3 array. `map` must be a valid MATLAB colormap. See `colormap` for a list of built-in MATLAB colormaps. Most image file formats do not support colormaps with more than 256 entries.

Example: `[0,0,0;0.5,0.5,0.5;1,1,1]`

Example: `jet(60)`

Data Types: `double`

`fmt` — Format of output file
string

Format of the output file, specified as one of the following strings.

This table also summarizes the types of images that `imwrite` can write. The MATLAB file format registry determines which file formats are supported. See `imformats` for more information about this registry.

For certain formats, `imwrite` can accept additional name-value pair arguments. To view these arguments, click the linked format names below.

Value of `fmt`	Format of Output File	Description
`'bmp'`	Windows® Bitmap (BMP)	1-bit, 8-bit, and 24-bit uncompressed images
`'gif'`	Graphics Interchange Format (GIF)	8-bit images
`'hdf'`	Hierarchical Data Format (HDF4)	8-bit raster image data sets with or without associated colormap, 24-bit raster image data sets
`'jpg'` or `'jpeg'`	Joint Photographic Experts Group (JPEG)	8-bit, 12-bit, and 16-bit Baseline JPEG images **Note:** `imwrite` converts indexed images to RGB before writing data to JPEG files, because the JPEG format does not support indexed images.

`'jp2'` or `'jpx'`	JPEG 2000 — Joint Photographic Experts Group 2000	1-bit, 8-bit, and 16-bit JPEG 2000 images
`'pbm'`	Portable Bitmap (PBM)	Any 1-bit PBM image, ASCII (plain) or raw (binary) encoding
`'pcx'`	Windows Paintbrush (PCX)	8-bit images
`'pgm'`	Portable Graymap (PGM)	Any standard PGM image; ASCII (plain) encoded with arbitrary color depth; raw (binary) encoded with up to 16 bits per gray value
`'png'`	Portable Network Graphics (PNG)	1-bit, 2-bit, 4-bit, 8-bit, and 16-bit grayscale images; 8-bit and 16-bit grayscale images with alpha channels; 1-bit, 2-bit, 4-bit, and 8-bit indexed images; 24-bit and 48-bit truecolor images; 24-bit and 48-bit truecolor images with alpha channels
`'pnm'`	Portable Anymap (PNM)	Any of the PPM/PGM/PBM formats, chosen automatically
`'ppm'`	Portable Pixmap (PPM)	Any standard PPM image: ASCII (plain) encoded with arbitrary color depth or raw (binary) encoded with up to 16 bits per color component
`'ras'`	Sun™ Raster (RAS)	Any RAS image, including 1-bit bitmap, 8-bit indexed, 24-bit truecolor, and 32-bit truecolor with alpha
`'tif'` or `'tiff'`	Tagged Image File Format (TIFF)	Baseline TIFF images, including: • 1-bit, 8-bit, 16-bit, 24-bit, and 48-bit uncompressed images and images with packbits, LZW, or Deflate compression • 1-bit images with CCITT 1D, Group 3, and Group 4 compression • CIELAB, ICCLAB, and CMYK images
`'xwd'`	X Windows Dump (XWD)	8-bit ZPixmaps

Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside single quotes (`' '`). You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example: `imwrite(A,'myFile.png','BitDepth',8)` writes the data in `A` using 8 bits to represent each pixel.

GIF — Graphics Interchange Format

`'BackgroundColor'` — Color to use as background color
scalar integer

Color to use as background color for the indexed image, specified as the comma-separated pair consisting of `'BackgroundColor'` and a scalar integer corresponding to the colormap index.

The background color is used for some disposal methods in animated GIFs.

- If image data `A` is `uint8` or `logical`, then the colormap index is zero-based.
- If image data `A` is `double`, then the colormap index is one-based.

The default background color corresponds to the first color in the colormap.

Example: `'BackgroundColor',15`

`'Comment'` — Comment to add to image
string | cell array of strings

Comment to add to the image, specified as the comma-separated pair consisting of `'Comment'` and a string or a 1-by-`n` cell array of strings. For a cell array of strings, `imwrite` adds a carriage return after each string.

Example: `'Comment',{'Sample #314','January 5, 2013'}`

Data Types: `char` | `cell`

`'DelayTime'` — Delay before displaying next image
0.5 (default) | scalar value in the range [0,655]

Delay before displaying next image, in seconds, specified as the comma-separated pair consisting of `'DelayTime'` and a scalar value in the range [0,655]. A value of `0` displays images as fast as your hardware allows.

Example: `'DelayTime',60`

`'DisposalMethod'` — Disposal method of animated GIF
`'doNotSpecify'` (default) | `'leaveInPlace'` | `'restoreBG'` | `'restorePrevious'`

Disposal method of an animated GIF, specified as the comma-separated pair consisting of `'DisposalMethod'` and one of the following strings.

Value of `DisposalMethod`	Result
`'doNotSpecify'` (default)	Replace one full-size, nontransparent frame with another.
`'leaveInPlace'`	Any pixels not covered up by the next frame continue to display.
`'restoreBG'`	The background color or background tile shows through transparent pixels.
`'restorePrevious'`	Restore to the state of a previous, undisposed frame.

Example: `'DisposalMethod','restoreBG'`

`'Location'` — Offset of screen relative to image
`[0,0]` (default) | two-element vector

Offset of the screen relative to the image, measured from the top left corner of each, specified as the comma-separated pair consisting of `'Location'` and a two-element vector. The first vector element specifies the offset from the top, and the second element specifies the offset from the left, in pixels.

Example: `'Location',[10,15]`

Data Types: `double`

`'LoopCount'` — Number of times to repeat animation
`Inf` (default) | integer in the range `[0,65535]`

Number of times to repeat the animation, specified as the comma-separated pair consisting of `'LoopCount'` and either an integer in the range `[0,65535]`, or the value `Inf`. If you specify 0, the animation plays once. If you specify the value 1, the animation plays twice, and so on. A `LoopCount` value of `Inf` causes the animation to continuously loop.

To enable animation within Microsoft® PowerPoint®, specify a value for `'LoopCount'` within the range [1,65535]. Some Microsoft applications interpret the value 0 to mean do not loop at all.

Example: `'LoopCount',3`

`'ScreenSize'` — Height and width of frame
height and width of input image (default) | two-element vector

Height and width of the frame, specified as the comma-separated pair consisting of `'ScreenSize'` and a two-element vector. When you use the `ScreenSize` argument with `'Location'`, it provides a way to write frames to the image that are smaller than the whole frame. `'DisposalMethod'` determines the fill value for pixels outside the frame.

Example: `'ScreenSize',[1000 1060]`

Data Types: `double`

`'TransparentColor'` — Color to use as transparent color
scalar integer

Color to use as transparent color for the image, specified as the comma-separated pair consisting of `'TransparentColor'` and a scalar integer corresponding to the colormap index.

- If image data `A` is `uint8` or `logical`, then indexing begins at `0`.
- If image data `A` is `double`, then indexing begins at `1`.

Example: `'TransparentColor',20`

`'WriteMode'` — Writing mode
`'overwrite'` (default) | `'append'`

Writing mode, specified as the comma-separated pair consisting of `'WriteMode'` and either `'overwrite'` or `'append'`. In `overwrite` mode, `imwrite` overwrites an existing file,`filename`. In `append` mode, `imwrite` adds a single frame to the existing file.

Example: `'WriteMode','append'`

HDF4 — Hierarchical Data Format

`'Compression'` — Compression scheme
`'none'` (default) | `'jpeg'` | `'rle'`

Compression scheme, specified as the comma-separated pair consisting of `'Compression'` and one of the following strings.

Value of `Compression`	Result
`'none'` (default)	No compression
`'jpeg'`	JPEG compression. Valid only for grayscale and RGB images.
`'rle'`	Run-length encoding. Valid only for grayscale and indexed images.

Example: `'Compression','jpeg'`

`'Quality'` — Quality of JPEG-compressed file

75 (default) | scalar in the range [0,100]

Quality of the JPEG-compressed file, specified as the comma-separated pair consisting of `'Quality'` and a scalar in the range [0,100], where 0 is lower quality and higher compression, and 100 is higher quality and lower compression. This parameter applies only if `'Compression'` is `'jpeg'`.

Example: `'Quality',25`

`'WriteMode'` — Writing mode
`'overwrite'` (default) | `'append'`

Writing mode, specified as the comma-separated pair consisting of `'WriteMode'` and either `'overwrite'` or `'append'`. In `overwrite` mode, `imwrite` overwrites an existing file,`filename`. In `append` mode, `imwrite` adds a single frame to the existing file.

Example: `'WriteMode','append'`

JPEG — Joint Photographic Experts Group

`'BitDepth'` — Number of bits per pixel
8 (default) | scalar

Number of bits per pixel, specified as the comma-separated pair consisting of `'BitDepth'` and a scalar.

- For grayscale images, the `BitDepth` value can be 8, 12, or 16. The default value is 8. For 16-bit images, the `'Mode'` name-value pair argument must be `'lossless'`.
- For color images, the `BitDepth` value is the number of bits per plane, and can be 8 or 12. The default is 8 bits per plane.

Example: `'BitDepth',12`

`'Comment'` — Comment to add to image
string | character array | n-by-1 cell array of strings

Comment to add to the image, specified as the comma-separated pair consisting of `'Comment'` and a single string, a character array, or an n-by-1 cell array of strings. `imwrite` writes each row of input as a comment in the JPEG file.

Example: `'Comment',{'First line';'second line';'third line'}`

Data Types: `char` | `cell`

`'Mode'` — Type of compression
`'lossy'` (default) | `'lossless'`

Type of compression, specified as the comma-separated pair consisting of `'Mode'` and one of the following strings:

- `'lossy'`
- `'lossless'`

Example: `'Mode','lossless'`

`'Quality'` — Quality of output file
75 (default) | scalar in the range [0,100]

Quality of the output file, specified as the comma-separated pair consisting of `'Quality'` and a scalar in the range [0,100], where 0 is lower quality and higher compression, and 100 is higher quality and lower compression. A `Quality` value of 100 does not write a lossless JPEG image. Instead, use the `'Mode','lossless'` name-value pair argument.

Example: `'Quality',100`

JPEG 2000— Joint Photographic Experts Group 2000

`'Comment'` — Comment to add to image
string | character array | cell array of strings

Comment to add to the image, specified as the comma-separated pair consisting of `'Comment'` and a single string, a character array, or a cell array of strings. `imwrite` writes each row of input as a comment in the JPEG 2000 file.

Example: `'Comment',{'First line';'second line';'third line'}`

Example: `'Comment',{'First line','second line','third line'}`

Data Types: `cell` | `char`

`'CompressionRatio'` — Target compression ratio
1 (default) | scalar

Target compression ratio, specified as the comma-separated pair consisting of `'CompressionRatio'` and a real scalar greater than or equal to 1. The compression ratio is the ratio of the input image size to the output compressed size. For example, a value of 2.0 implies that the output image size is half of the input image size or less. A higher value implies a smaller file size and reduced image quality. The compression ratio does not take into account the header size.

Specifying `CompressionRatio` is valid only when `'Mode'` is `'lossy'`.

Example: `'CompressionRatio',3`

`'Mode'` — Type of compression
`'lossy'` (default) | `'lossless'`

Type of compression, specified as the comma-separated pair consisting of `'Mode'` and one of the following strings:

- `'lossy'`
- `'lossless'`

Example: `'Mode','lossless'`

`'ProgressionOrder'` — Order of packets in code stream
`'LRCP'` (default) | `'RLCP'` | `'RPCL'` | `'PCRL'` | `'CPRL'`

Order of packets in the code stream, specified as the comma-separated pair consisting of `'ProgressionOrder'` and one of the following strings:

- `'LRCP'`
- `'RLCP'`
- `'RPCL'`
- `'PCRL'`
- `'CPRL'`

The characters in the text strings represent the following: `L` = layer, `R` = resolution, `C` = component and `P` = position.

Example: `'ProgressionOrder','RLCP'`

`'QualityLayers'` — Number of quality layers
1 (default) | integer in the range [1,20]

Number of quality layers, specified as the comma-separated pair consisting of `'QualityLayers'` and an integer in the range [1,20].

Example: `'QualityLayers',8`

`'ReductionLevels'` — Number of reduction levels
4 (default) | integer in the range [1,8]

Number of reduction levels, or wavelet decomposition levels, specified as the comma-separated pair consisting of `'ReductionLevels'` and an integer in the range [1,8].

Example: `'ReductionLevels',6`

`'TileSize'` — Tile height and width
image size (default) | two-element vector

Tile height and width, specified as the comma-separated pair consisting of `'TileSize'` and a two-element vector. The minimum size you can specify is `[128 128]`.

Example: `'TileSize',[130 130]`

PBM-, PGM-, and PPM — Portable Bitmap, Graymap, Pixmap

`'Encoding'` — Encoding
`'rawbits'` (default) | `'ASCII'`

Encoding, specified as the comma-separated pair consisting of `'Encoding'` and either `'rawbits'` for binary encoding, or `'ASCII'` for plain encoding.

Example: `'Encoding','ASCII'`

`'MaxValue'` — Maximum gray or color value
scalar

Maximum gray or color value, specified as the comma-separated pair consisting of `'MaxValue'` and a scalar.

Available only for PGM and PPM files. For PBM files, this value is always 1.

If the image array is `uint16`, then the default value for `MaxValue` is `65535`. Otherwise, the default value is `255`.

Example: `'MaxValue',510`

PNG — Portable Network Graphics

In addition to the following name-value pair arguments, you can use any parameter name that satisfies the PNG specification for keywords. That is, the name uses only printable characters, contains 80 or fewer characters, and does not contain leading or trailing spaces. The value corresponding to these user-specified names must be a string that contains no control characters other than linefeed.

'Alpha' — Transparency of each pixel
matrix of values in the range [0,1]

Transparency of each pixel, specified as the comma-separated pair consisting of `'Alpha'` and a matrix of values in the range [0,1]. The row and column dimensions of the `Alpha` matrix must be the same as those of the image data array. You can specify `Alpha` only for grayscale (`m`-by-`n`) and truecolor (`m`-by-`n`-by-3) image data.

> **Note:** You cannot specify both `'Alpha'` and `'Transparency'` at the same time.

Data Types: `double` | `uint8` | `uint16`

'Author' — Author information
string

Author information, specified as the comma-separated pair consisting of `'Author'` and a string.

Example: "`Author','Ann Smith'`

Data Types: `char`

'Background' — Background color when compositing transparent pixels
scalar in the range `[0,1]` | integer in the range `[1,P]` | 3-element vector in the range `[0,1]`

Background color when compositing transparent pixels, specified as the comma-separated pair consisting of `'Background'` and a value dependent on the image data, as follows.

Image Type	Form of `Background` Value
Grayscale images	Scalar in the range `[0,1]`.
Indexed images	Integer in the range `[1,P]`, where `P` is the colormap length. For example, `'Background',50` sets the background color to the color specified by the 50th index in the colormap.
Truecolor images	Three-element vector of RGB intensities in the range `[0,1]`. For example, `'Background',[0 1 1]` sets the background color to cyan.

Data Types: `double`

'BitDepth' — Number of bits per pixel
scalar

Number of bits per pixel, specified as the comma-separated pair consisting of `'BitDepth'` and a scalar. Depending on the output image, the scalar can be one of the following values.

Image Type	Allowed Values for `BitDepth`
Grayscale images	1, 2, 4, 8, or 16
Grayscale images with an alpha channel	8 or 16
Indexed images	1, 2, 4, or 8
Truecolor images	8 or 16

- If the image is of class `double` or `uint8`, then the default bit depth is 8 bits per pixel.
- If the image is `uint16`, then the default is 16 bits per pixel.

- If the image is `logical`, then the default is 1 bit per pixel.

Example: `'BitDepth',4`

`'Chromaticities'` — Reference white point and primary chromaticities
8-element vector

Reference white point and primary chromaticities, specified as the comma-separated pair consisting of `'Chromaticities'` and an 8-element vector, `[wx wy rx ry gx gy bx by]`. The elements `wx` and `wy` are the chromaticity coordinates of the white point, and the elements `rx`, `ry`, `gx`, `gy`, `bx`, and `by` are the chromaticity coordinates of the three primary colors.

If you specify `Chromaticities`, you should also specify the `Gamma` name-value pair argument.

Example: `'Chromaticities',[0.312,0.329,0.002,0.002,0.001,0.001,0.115,0.312]`

Data Types: `double`

`'Comment'` — Comment to add to image
string

Comment to add to the image, specified as the comma-separated pair consisting of `'Comment'` and a string.

`'Copyright'` — Copyright notice
string

Copyright notice, specified as the comma-separated pair consisting of `'Copyright'` and a string.

`'CreationTime'` — Time of original image creation
string

Time of original image creation, specified as a string.

`'Description'` — Description of image
string

Description of the image, specified as the comma-separated pair consisting of `'Description'` and a string.

`'Disclaimer'` — Legal disclaimer
string

Legal disclaimer, specified as the comma-separated pair consisting of `'Disclaimer'` and a string.

`'Gamma'` — File gamma
scalar

File gamma, specified as the comma-separated pair consisting of `'Gamma'` and a scalar.

Example: `'Gamma',2.2`

`'ImageModTime'` — Time of last image modification
serial date number | date string

Time of the last image modification, specified as the comma-separated pair consisting of `'ImageModTime'` and a MATLAB serial date number or a date string that can be converted to a date vector using the `datevec` function. Values should be in Coordinated Universal Time (UTC).

The default `ImageModTime` value is the time when you call `imwrite`.

Example: `'ImageModTime','17-Jan-2013 11:23:10'`

Data Types: `double` | `char`

`'InterlaceType'` — Interlacing scheme
`'none'` (default) | `'adam7'`

Interlacing scheme, specified as the comma-separated pair consisting of `'InterlaceType'` and either `'none'` for no interlacing, or `'adam7'` to use the Adam7 algorithm.

Example: `'InterlaceType','adam7'`

`'ResolutionUnit'` — Unit for image resolution
`'unknown'` (default) | `'meter'`

Unit for image resolution, specified as the comma-separated pair consisting of `'ResolutionUnit'` and either `'unknown'` or `'meter'`. If you specify `ResolutionUnit`, you must include at least one of the `XResolution` and `YResolution` name-value pair arguments. When the value of `ResolutionUnit` is `'meter'`, the `XResolution` and `YResolution` values are interpreted in pixels per meter.

Example: `'ResolutionUnit','meter','XResolution',1000`

`'SignificantBits'` — Number of bits to regard as significant
`[]` (default) | scalar | vector

Number of bits in the data array to regard as significant, specified as the comma-separated pair consisting of `'SignificantBits'` and a scalar or a vector in the range [1,`BitDepth`]. Depending on the output image type, the value must be in the following form.

Image Type	Form of `SignificantBits` Value
Grayscale image without an alpha channel	Scalar
Grayscale image with an alpha channel	2-element vector
Indexed image	3-element vector
Truecolor image without an alpha channel	3-element vector
Truecolor image with an alpha channel	4-element vector

Example: `'SignificantBits',[2,3]`

`'Software'` — Software used to create the image
string

Software used to create the image, specified as the comma-separated pair consisting of `'Software'` and a string.

`'Source'` — Device used to create the image
string

Device used to create the image, specified as the comma-separated pair consisting of `'Source'` and a string.

`'Transparency'` — Pixels to consider transparent
[ ] (default) | scalar in the range [0,1] | vector

Pixels to consider transparent when no alpha channel is used, specified as the comma-separated pair consisting of `'Transparency'` and a scalar or a vector. Depending on the output image, the value must be in the following form.

Image Type	**Form of `Transparency` Value**
Grayscale images	Scalar in the range [0,1], indicating the grayscale color to be considered transparent.
Indexed images	`Q`-element vector of values in the range [0,1], where `Q` is no larger than the colormap length and each value indicates the transparency associated with the corresponding colormap entry. In most cases, `Q = 1`.
Truecolor images	3-element vector of RGB intensities in the range [0,1], indicating the truecolor color to consider transparent.

Note: You cannot specify both `'Transparency'` and `'Alpha'` at the same time.

Example: `'Transparency',[1 1 1]`

Data Types: `double`

`'Warning'` — Warning of nature of content
string

Warning of nature of content, specified as the comma-separated pair consisting of `'Warning'` and a string.

`'XResolution'` — Image resolution in horizontal direction
scalar

Image resolution in the horizontal direction, in pixels/unit, specified as the comma-separated pair consisting of `'XResolution'` and a scalar. Define the unit by specifying the `ResolutionUnit` name-value pair argument.

If you do not also specify `YResolution`, then the `XResolution` value applies to both the horizontal and vertical directions.

Example: `'XResolution',900`

`'YResolution'` — Image resolution in vertical direction
scalar

Image resolution in the vertical direction, in pixels/unit, specified as the comma-separated pair consisting of `'XResolution'` and a scalar. Define the unit by specifying the `ResolutionUnit` name-value pair argument.

If you do not also specify `XResolution`, then the `YResolution` value applies to both the horizontal and vertical directions.

Example: `'YResolution',900`

RAS — Sun Raster Graphic

`'Alpha'` — Transparency of each pixel
[ ] (default) | matrix

Transparency of each pixel, specified as the comma-separated pair consisting of `'Alpha'` and a matrix with row and column dimensions the same as those of the image data array.

Valid only for truecolor (`m`-by-`n`-by-3) image data.

Data Types: `double` | `single` | `uint8` | `uint16`

`'Type'` — Image type
`'standard'` (default) | `'rgb'` | `'rle'`

Image type, specified as the comma-separated pair consisting of `'Type'` and one of the following strings.

Value of `Type`	Description
`'standard'` (default)	Uncompressed, B-G-R color order for truecolor images
`'rgb'`	Uncompressed, R-G-B color order for truecolor images
`'rle`	Run-length encoding of 1-bit and 8-bit images

Example: `'Type','rgb'`

TIFF — Tagged Image File Format

`'ColorSpace'` — Color space representing color data
`'rgb'` (default) | `'cielab'` | `'icclab'`

Color space representing the color data, specified as the comma-separated pair consisting of `'ColorSpace'` and one of the following strings:

- `'rgb'`
- `'cielab'`
- `'icclab'`

Valid only when the image data array, `A`, is truecolor (`m`-by-`n`-by-3). To use the CMYK color space in a TIFF file, do not use the `'ColorSpace'` name-value pair argument. Instead, specify an `m`-by-`n`-by-4 image data array.

`imwrite` can write color image data that uses the *L*a*b** color space to TIFF files. The 1976 CIE *L*a*b** specification defines numeric values that represent luminance (*L**) and chrominance (*a** and *b**) information. To store *L*a*b** color data in a TIFF file, the values must be encoded to fit into either 8-bit or 16-bit storage. `imwrite` can store *L*a*b** color data in a TIFF file using the following encodings:

- CIELAB encodings — 8-bit and 16-bit encodings defined by the TIFF specification
- ICCLAB encodings — 8-bit and 16-bit encodings defined by the International Color Consortium

The output class and encoding used by `imwrite` depends on the class of the input image data array and the `ColorSpace` value, as shown in the following table. (The 8-bit and 16-bit CIELAB encodings cannot be input arrays because they use a mixture of signed and unsigned values and cannot be represented as a single MATLAB array.)

Input Class and Encoding	Value of `ColorSpace`	Output Class and Encoding
8-bit ICCLAB	`'icclab'`	8-bit ICCLAB

Values are integers in the range [0 255]. L^* values are multiplied by `255/100`. `128` is added to both the a^* and b^* values.		
	`'cielab'`	8-bit CIELAB
16-bit ICCLAB Values are integers in the range [0, 65280]. L^* values are multiplied by `65280/100`. `32768` is added to both the a^* and b^* values, which are represented as integers in the range [0,65535].	`'icclab'`	16-bit ICCLAB
	`'cielab'`	16-bit CIELAB
Double-precision 1976 CIE $L^*a^*b^*$ values L^* is in the dynamic range [0, 100]. a^* and b^* can take any value. Setting a^* and b^* to 0 (zero) produces a neutral color (gray).	`'icclab'`	8-bit ICCLAB
	`'cielab'`	8-bit CIELAB

Example: `'ColorSpace','cielab'`

`'Compression'` — Compression scheme

`'packbits'` | `'none'` | `'lzw'` | `'deflate'` | `'jpeg'` | `'ccitt'` | `'fax3'` | `'fax4'`

Compression scheme, specified as the comma-separated pair consisting of `'Compression'` and one of the following strings:

- `'packbits'` (default for nonbinary images)
- `'none'`
- `'lzw'`
- `'deflate'`
- `'jpeg'`
- `'ccitt'` (binary images only, and the default for such images)
- `'fax3'` (binary images only)
- `'fax4'` (binary images only)

`'jpeg'` is a lossy compression scheme; other compression modes are lossless. Also, if you specify `'jpeg'` compression, you must specify the `'RowsPerStrip'` parameter and the value must be a multiple of 8.

Example: `'Compression','none'`

`'Description'` — Image description

string

Image description, specified by the comma-separated pair consisting of `'Description'` and a string. This is the text that `imfinfo` returns in the `ImageDescription` field for the output image.

Example: `'Description','Sample 2A301'`

`'Resolution'` — X- and Y-resolution
72 (default) | scalar | two-element vector

X- and Y-resolution, specified as the comma-separated pair consisting of `'Resolution'` and a scalar indicating both resolution, or a two-element vector containing the X-Resolution and Y-Resolution.

Example: `'Resolution',80`

Example: `'Resolution',[320,72]`

Data Types: `double`

`'RowsPerStrip'` — Number of rows to include in each strip
scalar

Number of rows to include in each strip, specified as the comma-separated pair consisting of `'RowsPerStrip'` and a scalar. The default value is such that each strip is about 8 kilobytes.

You must specify `RowsPerStrip` if you specify `'jpeg'` compression. The value must be a multiple of 8.

Example: `'RowsPerStrip',16`

Data Types: `double` | `single` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

`'WriteMode'` — Writing mode
`'overwrite'` (default) | `'append'`

Writing mode, specified as the comma-separated pair consisting of `'WriteMode'` and either `'overwrite'` or `'append'`. In `overwrite` mode, `imwrite` overwrites an existing file. In `append` mode, `imwrite` adds a page to the existing file.

Example: `'WriteMode','append'`

More About

collapse all

Tips

- For copyright information, see the `libtiffcopyright.txt` file.

See Also

`fwrite` | `getframe` | `imfinfo` | `imformats` | `imread` | `Tiff`

Introduced before R2006a