Converting Color Data Between Color Spaces

On this page…

Understanding Color Spaces and Color Space Conversion

Converting Between Device-Independent Color Spaces

Performing Profile-Based Color Space Conversions

Converting Between Device-Dependent Color Spaces

Understanding Color Spaces and Color Space Conversion

The Image Processing Toolbox™ software represents colors as RGB values, either directly (in an RGB image) or indirectly (in an indexed image, where the colormap is stored in RGB format). However, there are other models besides RGB for representing colors numerically. The various models are referred to as *color spaces* because most of them can be mapped into a 2-D, 3-D, or 4-D coordinate system; thus, a color specification is made up of coordinates in a 2-D, 3-D, or 4-D space.

The various color spaces exist because they present color information in ways that make certain calculations more convenient or because they provide a way to identify colors that is more intuitive. For example, the RGB color space defines a color as the percentages of red, green, and blue hues mixed together. Other color models describe colors by their hue (green), saturation (dark green), and luminance, or intensity.

The toolbox supports these color spaces by providing a means for converting color data from one color space to another through a mathematical transformation.

Converting Between Device-Independent Color Spaces

The standard terms used to describe colors, such as hue, brightness, and intensity, are subjective and make comparisons difficult.

In 1931, the International Commission on Illumination, known by the acronym CIE, for *Commission Internationale de l'Éclairage*, studied human color perception and developed a standard, called the CIE *XYZ*. This standard defined a three-dimensional space where three values, called tristimulus values, define a color. This standard is still widely used today.

In the decades since that initial specification, the CIE has developed several additional color space specifications that attempt to provide alternative color representations that are better suited to some purposes than XYZ. For example, in 1976, in an effort to get a perceptually uniform color space that could be correlated with the visual appearance of colors, the CIE created the *L*a*b** color space.

The toolbox supports conversions between members of the CIE family of device-independent color spaces. In addition, the toolbox also supports conversions between these CIE color spaces and the sRGB color space. This color space was defined by an industry group to describe the characteristics of a typical PC monitor.

This section

- Lists the supported device-independent color spaces
- Provides an example of how to perform a conversion
- Provides guidelines about data type support of the various conversions

Supported Conversions

This table lists all the device-independent color spaces that the toolbox supports.

Color Space	Description	Supported Conversions
XYZ	The original, 1931 CIE color space specification.	*xyY*, *uvl*, *u'v'L*, and *L*a*b**
xyY	CIE specification that provides normalized chromaticity values. The capital *Y* value represents luminance and is the same as in *XYZ*.	*XYZ*
uvL	CIE specification that attempts to make the chromaticity plane more visually uniform. *L* is luminance and is the same as *Y* in *XYZ*.	*XYZ*
u'v'L	CIE specification in which *u* and *v* are rescaled to improve uniformity.	*XYZ*
L*a*b*	CIE specification that attempts to make the luminance scale more perceptually uniform. *L** is a nonlinear scaling of *L*, normalized to a reference white point.	*XYZ*
L*ch	CIE specification where *c* is chroma and *h* is hue. These values are a polar coordinate conversion of *a** and *b** in *L*a*b**.	*L*a*b**
sRGB	Standard adopted by major manufacturers that characterizes the average PC monitor.	*XYZ* and *L*a*b**

Example: Performing a Color Space Conversion

To illustrate a conversion between two device-independent color spaces, this example reads an RGB color image into the MATLAB® workspace and converts the color data to the XYZ color space:

1. Import color space data. This example reads an RGB color image into the MATLAB workspace.

   ```
   I_rgb = imread('peppers.png');
   ```

2. Create a color transformation structure. A color transformation structure defines the conversion between two color spaces. You use the `makecform` function to create the structure, specifying a transformation type string as an argument.

   This example creates a color transformation structure that defines a conversion from RGB color data to *XYZ* color data.

   ```
   C = makecform('srgb2xyz');
   ```

3. Perform the conversion. You use the `applycform` function to perform the conversion, specifying as arguments the color data you want to convert and the color transformation structure that defines the conversion. The `applycform` function returns the converted data.

   ```
   I_xyz = applycform(I_rgb,C);
   ```

   View a list of the workspace variables for the original and converted images.

   ```
   whos
   ```

text

- **Understanding Device Profiles**
- **Reading ICC Profiles**
- **Writing Profile Information to a File**
- **Example: Performing a Profile-Based Conversion**
- **Specifying the Rendering Intent**

Understanding Device Profiles

If two colors have the same CIE colorimetry, they will match *if viewed under the same conditions*. However, because color images are typically produced for a wide variety of viewing environments, it is necessary to go beyond simple application of the CIE system.

For this reason, the International Color Consortium (ICC) has defined a Color Management System (CMS) that provides a means for communicating color information among input, output, and display devices. The CMS uses device *profiles* that contain color information specific to a particular device. Vendors that support CMS provide profiles that characterize the color reproduction of their devices, and methods, called Color Management Modules (CMM), that interpret the contents of each profile and perform the necessary image processing.

Device profiles contain the information that color management systems need to translate color data between devices. Any conversion between color spaces is a mathematical transformation from some domain space to a range space. With profile-based conversions, the domain space is often called the *source space* and the range space is called the *destination space*. In the ICC color management model, profiles are used to represent the source and destination spaces.

For more information about color management systems, go to the International Color Consortium Web site, `www.color.org` **(http://www.color.org)** .

Reading ICC Profiles

To read an ICC profile into the MATLAB workspace, use the `iccread` function. In this example, the function reads in the profile for the color space that describes color monitors.

```
P = iccread('sRGB.icm');
```

You can use the `iccfind` function to find ICC color profiles on your system, or to find a particular ICC color profile whose description contains a certain text string. To get the name of the directory that is the default system repository for ICC profiles, use `iccroot`.

`iccread` returns the contents of the profile in the structure `P`. All profiles contain a header, a tag table, and a series of tagged elements. The header contains general information about the profile, such as the device class, the device color space, and the file size. The tagged elements, or tags, are the data constructs that contain the information used by the CMM. For more information about the contents of this structure, see the `iccread` function reference page.

Using `iccread`, you can read both Version 2 (ICC.1:2001-04) or Version 4 (ICC.1:2001-12) ICC profile formats. For detailed information about these specifications and their differences, visit the ICC web site, `www.color.org` **(http://www.color.org)** .

Writing Profile Information to a File

To export ICC profile information from the MATLAB workspace to a file, use the `iccwrite` function. This example reads a profile into the MATLAB workspace and then writes the profile information out to a new file.

```
P = iccread('sRGB.icm');
P_new = iccwrite(P,'my_profile.icm');
```

`iccwrite` returns the profile it writes to the file in `P_new` because it can be different than the input profile `P`. For example, `iccwrite` updates the Filename field in `P` to match the name of the file specified as the second argument.

`iccwrite` checks the validity of the input profile structure. If any required fields are missing, `iccwrite` returns an error message. For more information about the writing ICC profile data to a file, see the **iccwrite** function reference page. To determine if a structure is a valid ICC profile, use the **isicc** function.

Using `iccwrite`, you can export profile information in both Version 2 (ICC.1:2001-04) or Version 4 (ICC.1:2001-12) ICC profile formats. The value of the `Version` field in the file profile header determines the format version. For detailed information about these specifications and their differences, visit the ICC web site, **www.color.org (http://www.color.org)** .

Example: Performing a Profile-Based Conversion

To illustrate a profile-based color space conversion, this section presents an example that converts color data from the RGB space of a monitor to the CMYK space of a printer. This conversion requires two profiles: a monitor profile and a printer profile. The source color space in this example is monitor RGB and the destination color space is printer CMYK:

1. Import RGB color space data. This example imports an RGB color image into the MATLAB workspace.

   ```
   I_rgb = imread('peppers.png');
   ```

2. Read ICC profiles. Read the source and destination profiles into the MATLAB workspace. This example uses the sRGB profile as the source profile. The sRGB profile is an industry-standard color space that describes a color monitor.

   ```
   inprof = iccread('sRGB.icm');
   ```

   For the destination profile, the example uses a profile that describes a particular color printer. The printer vendor supplies this profile. (The following profile and several other useful profiles can be obtained as downloads from `www.adobe.com`.)

   ```
   outprof = iccread('USSheetfedCoated.icc');
   ```

3. Create a color transformation structure. You must create a color transformation structure to define the conversion between the color spaces in the profiles. You use the `makecform` function to create the structure, specifying a transformation type string as an argument.

   > **Note**   The color space conversion might involve an intermediate conversion into a device-independent color space, called the Profile Connection Space (PCS), but this is transparent to the user.

   This example creates a color transformation structure that defines a conversion from RGB color data to CMYK color data.

   ```
   C = makecform('icc',inprof,outprof);
   ```

4. Perform the conversion. You use the `applycform` function to perform the conversion, specifying

as arguments the color data you want to convert and the color transformation structure that defines the conversion. The function returns the converted data.

```
I_cmyk = applycform(I_rgb,C);
```

5. Write the converted data to a file. To export the CMYK data, use the `imwrite` function, specifying the format as TIFF. If the format is TIFF and the data is an m-by-n-by-4 array, `imwrite` writes CMYK data to the file.

```
imwrite(I_cmyk,'pep_cmyk.tif','tif')
```

To verify that the CMYK data was written to the file, use imfinfo to get information about the file and look at the `PhotometricInterpretation` field.

```
info = imfinfo('pep_cmyk.tif');
info.PhotometricInterpretation
ans =
    'CMYK'
```

Specifying the Rendering Intent

For most devices, the range of reproducible colors is much smaller than the range of colors represented by the PCS. It is for this reason that four rendering intents (or gamut mapping techniques) are defined in the profile format. Each one has distinct aesthetic and color-accuracy tradeoffs.

When you create a profile-based color transformation structure, you can specify the rendering intent for the source as well as the destination profiles. For more information, see the **makecform** reference information.

Converting Between Device-Dependent Color Spaces

The toolbox includes functions that you can use to convert RGB data to several common device-dependent color spaces, and vice versa:

- YIQ
- YCbCr
- Hue, saturation, value (HSV)

YIQ Color Space

The National Television Systems Committee (NTSC) defines a color space known as YIQ. This color space is used in televisions in the United States. One of the main advantages of this format is that grayscale information is separated from color data, so the same signal can be used for both color and black and white sets.

In the NTSC color space, image data consists of three components: luminance (Y), hue (I), and saturation (Q). The first component, *luminance*, represents grayscale information, while the last two components make up *chrominance (color information)*.

The function **rgb2ntsc** converts colormaps or RGB images to the NTSC color space. **ntsc2rgb** performs the reverse operation.

For example, these commands convert an RGB image to NTSC format.

```
RGB = imread('peppers.png');
YIQ = rgb2ntsc(RGB);
```

Because luminance is one of the components of the NTSC format, the RGB to NTSC conversion is also useful for isolating the gray level information in an image. In fact, the toolbox functions `rgb2gray` and `ind2gray` use the `rgb2ntsc` function to extract the grayscale information from a color image.

For example, these commands are equivalent to calling `rgb2gray`.

```
YIQ = rgb2ntsc(RGB);
I = YIQ(:,:,1);
```

> **Note** In the YIQ color space, I is one of the two color components, not the grayscale component.

YCbCr Color Space

The YCbCr color space is widely used for digital video. In this format, luminance information is stored as a single component (Y), and chrominance information is stored as two color-difference components (Cb and Cr). Cb represents the difference between the blue component and a reference value. Cr represents the difference between the red component and a reference value. (YUV, another color space widely used for digital video, is very similar to YCbCr but not identical.)

YCbCr data can be double precision, but the color space is particularly well suited to `uint8` data. For `uint8` images, the data range for Y is [16, 235], and the range for Cb and Cr is [16, 240]. YCbCr leaves room at the top and bottom of the full `uint8` range so that additional (nonimage) information can be included in a video stream.

The function **rgb2ycbcr** converts colormaps or RGB images to the YCbCr color space. **ycbcr2rgb** performs the reverse operation.

For example, these commands convert an RGB image to YCbCr format.

```
RGB = imread('peppers.png');
YCBCR = rgb2ycbcr(RGB);
```
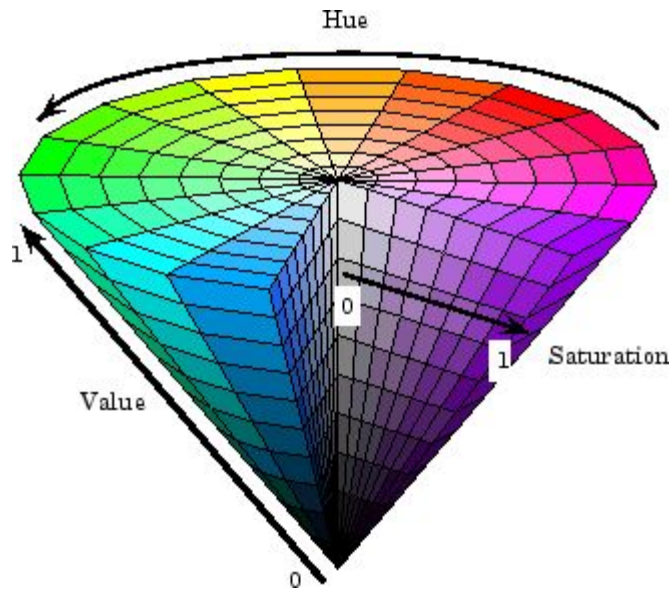
HSV Color Space

The HSV color space (Hue, Saturation, Value) is often used by people who are selecting colors (e.g., of paints or inks) from a color wheel or palette, because it corresponds better to how people experience color than the RGB color space does. The functions **rgb2hsv** and **hsv2rgb** convert images between the RGB and HSV color spaces.

> **Note:** MATLAB and the Image Processing Toolbox software do not support the HSI color space (Hue, Saturation, Intensity). However, if you want to work with color data in terms of hue, saturation, and intensity, the HSV color space is very similar. Another option is to use the LCH color space (Luminosity, Chroma, and Hue), which is a polar transformation of the CIE *L*a*b** color space — see **Converting Between Device-Independent Color Spaces**.

As hue varies from 0 to 1.0, the corresponding colors vary from red through yellow, green, cyan, blue, magenta, and back to red, so that there are actually red values both at 0 and 1.0. As saturation varies from 0 to 1.0, the corresponding colors (hues) vary from unsaturated (shades of gray) to fully saturated (no white component). As value, or brightness, varies from 0 to 1.0, the corresponding colors become increasingly brighter.

The following figure illustrates the HSV color space.

Illustration of the HSV Color Space

The `rgb2hsv` function converts colormaps or RGB images to the HSV color space. `hsv2rgb` performs the reverse operation. These commands convert an RGB image to the HSV color space.

```
RGB = imread('peppers.png');
HSV = rgb2hsv(RGB);
```

For closer inspection of the HSV color space, the next block of code displays the separate color planes (hue, saturation, and value) of an HSV image.

```
RGB=reshape(ones(64,1)*reshape(jet(64),1,192),[64,64,3]);
HSV=rgb2hsv(RGB);
H=HSV(:,:,1);
S=HSV(:,:,2);
V=HSV(:,:,3);
subplot(2,2,1), imshow(H)
subplot(2,2,2), imshow(S)
subplot(2,2,3), imshow(V)
subplot(2,2,4), imshow(RGB)
```

The Separated Color Planes of an HSV Image
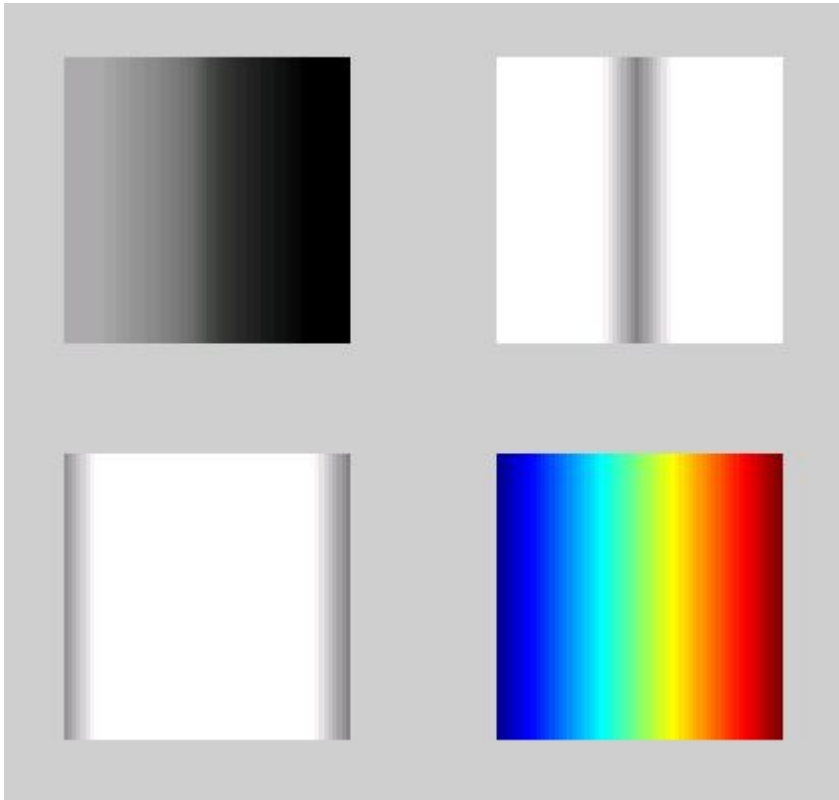
As the hue plane image in the preceding figure illustrates, hue values make a linear transition from high to low. If you compare the hue plane image against the original image, you can see that shades of deep blue have the highest values, and shades of deep red have the lowest values. (As stated previously, there are values of red on both ends of the hue scale. To avoid confusion, the sample image uses only the red values from the *beginning* of the hue range.)

Saturation can be thought of as the purity of a color. As the saturation plane image shows, the colors with the highest saturation have the highest values and are represented as white. In the center of the saturation image, notice the various shades of gray. These correspond to a mixture of colors; the cyans, greens, and yellow shades are mixtures of true colors. Value is roughly equivalent to brightness, and you will notice that the brightest areas of the value plane correspond to the brightest colors in the original image.