The conversion to gray is based on the YIQ standard where the gray output wave corresponds to the Y channel:

```
gray=0.299*red+0.587*green+0.114*blue.
```

If you wish to use a different set of transformation constants say $\{c_i\}$, you can perform the conversion on the command line:

```
gray2DWave=c1*image[p][q][0]+c2*image[p][q][1]+c3*image[p][q][2]
```

For large images this operation may be slow. A more efficient approach is:

```
Make/O/N=3 scaleWave={c1,c2,c3}
ImageTransform/D=scaleWave scalePlanes image // Creates M_ScaledPlanes
ImageTransform sumPlanes M_ScaledPlanes
```

In some applications it is desirable to extract information from the color of regions in the image. We therefore convert the image from RGB to the HSL color space and then perform operations on the first plane (hue) of the resulting 3D wave. In the following example we convert the RGB image wave peppers into HSL, extract the hue plane and produce a binary image in which the red hues are nonzero.

```
ImageTransform /U rgb2hsl peppers// Note the /U for unsigned short result
MatrixOp/O RedPlane=greater(5000,M_RGB2HSL[][][0])+greater(M_RGB2HSL[][][0],60000)
NewImage RedPlane          // Here white corresponds to red hues in the source
```

As you can see, the resulting image is binary, with white pixels corresponding to regions where the original image was predominantly red. The binary image can be used to discriminate between red and other hue regions. The second command line above converts hue values that range from 0 to 65535 to 1 if the color is in the "reddish" range, or zero if it is outside that range. The selection of values below 5000 is due to the fact that red hues appear on both sides of 0° (or 360°) of the hue circle.

Hue based image segmentation is also supported through the **ImageTransform** operation (see page V-417) using the **hslSegment**, **matchPlanes** or **selectColor** keywords. The same operation also supports color space conversions from RGB to CIE XYZ (D65 based) and from XYZ to RGB. See also **Handling Color** on page III-379 and **HSL Segmentation** on page III-374.

Grayscale or Value Transforms

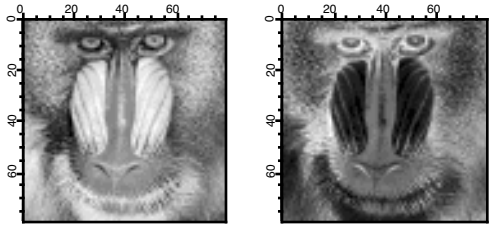
This class of transforms applies only to 2D waves or to individual layers of higher dimensional waves. They are called "grayscale" because 2D waves by themselves do not contain color information. We divide grayscale transforms into level mappings and mathematical operations.

Explicit Lookup Tables

Here is an example of using an explicit lookup table (LUT) to create the negative of an image the hard way:

```
Make/B/U/O/N=256 negativeLookup=255-x     // Create the lookup table
Duplicate/O baboon negativeBaboon
negativeBaboon=negativeLookup[negativeBaboon]// The lookup transformation
```

```
NewImage baboon
NewImage negativeBaboon
```



In this example the negativeBaboon image is a derived wave displayed with standard linear LUT. You can also obtain the same result using the original baboon image but displaying it with a negative LUT:

```
NewImage baboon
Make/N=256 negativeLookup=1-x/255   // Negative slope LUT from 1 to 0
ModifyImage baboon lookup=negativeLookup
```

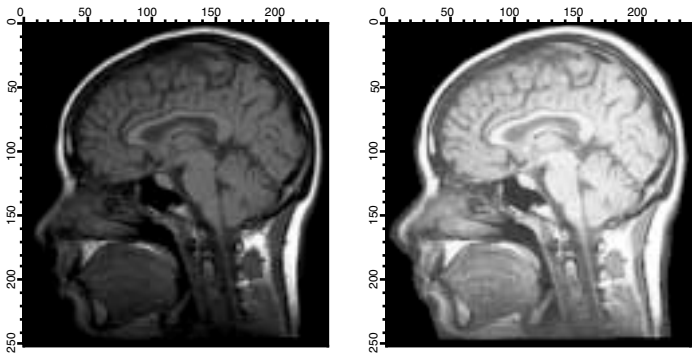
If you are willing to modify the original data you can execute:

```
ImageTransform invert baboon
```

Histogram Equalization

Histogram equalization maps the values of a grayscale image so that the resulting values utilize the entire available range of intensities:

```
NewImage MRI
ImageHistModification MRI
NewImage M_ImageHistEq
```



Adaptive Histogram Equalization

The **ImageHistModification** operation calculates a lookup table based on the cumulative histogram of the whole source image. The lookup table is then applied the output image. In cases where there are significant spatial variations in the histogram, a more local approach may be needed, i.e., perform the histogram equalization independently for different parts of the image and then combine the regional results by matching them across region boundaries. This is commonly referred to as "Adaptive Histogram Equalization".

```
ImageHistModification MRI
Duplicate/O M_ImageHistEq, globalHist
NewImage globalHist
ImageTransform/N={2,7} padImage MRI // To make the image divisible
ImageHistModification/A/C=10/H=2/V=2 M_paddedImage
NewImage M_ImageHistEq
```

The original image is 238 by 253 pixels. Because the number of rows and columns must be divisible by the number of equalization intervals, we first padded the image using the ImageTransform **padImage**. The result is an image that is 240 by 260. If you do not find the resulting adaptive histogram sufficiently different