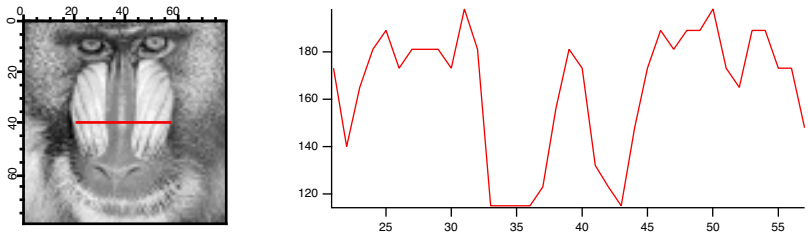
ImageLineProfile Operation

The **ImageLineProfile** operation (see page V-389) is somewhat of a misnomer as it samples the image along a path consisting of an arbitrary number of line segments. To use the operation you first need to create the description of the path using a pair of waves. Here is a simple example:

```
NewImage root:images:baboon    // Display the image that we want to profile
// Create the pair of waves representing a straight line path.
Make/O/N=2 xPoints={21,57}, yPoints={40,40}
AppendToGraph/T yPoints vs xPoints  // display the path on the image
// Calculate the profile.
ImageLineProfile xwave=xPoints, ywave=yPoints, srcwave=root:images:baboon
Display W_ImageLineProfile vs W_LineProfileX // display the profile
```



You can create a more complex path consisting of an arbitrary number of points. In this case you may want to take advantage of the W_LineProfileX and W_LineProfileY waves that the operation creates and plot the profile as a 3D path plot (see "Path Plots" in the Visualization help file). See also the IP Tutorial experiment for more elaborate examples.

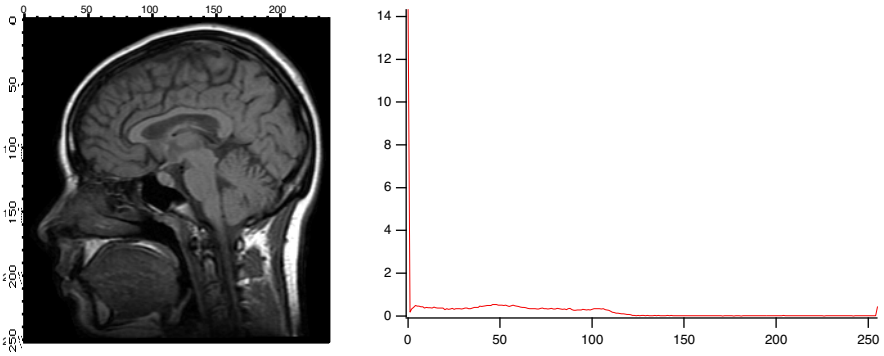
Note: If you are working with 3D waves with more than 3 layers, you can use `ImageLineProfile/P=`*`plane`* to specify the plane for which the profile is computed.

If you are using the line profile to extract a sequential array of data (a row or column) from the wave it is more efficient (about a factor of 3.5 in speed) to extract the data using ImageTransform **getRow** or **getCol**.

Histograms

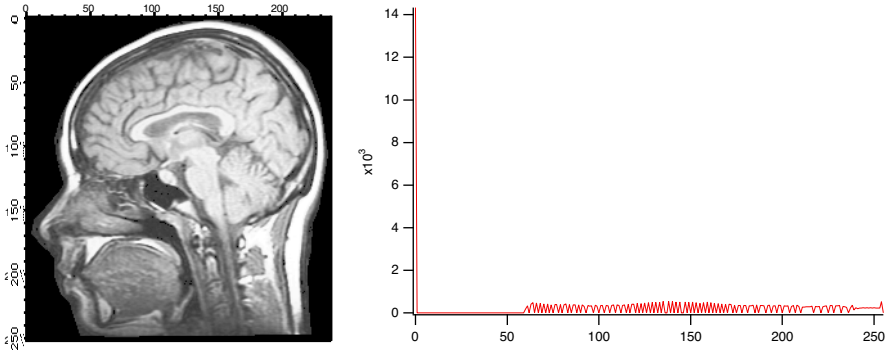
The histograms is a very important tool in image analysis. For example, the simplest approach to automating the detection of the background in an image is to calculate the histogram and to choose the pixel value which occurs with the highest frequency. Histograms are also very important in determining threshold values and in enhancing image contrast. Here are some examples using image histograms:

```
NewImage root:images:mri
ImageHistogram root:images:mri
Duplicate W_ImageHist origMriHist
Display /W=(201.6,45.2,411,223.4) origMriHist
```



It is obvious from the histogram that the image is rather dark and that the background is most likely zero. The small counts for pixels above 125 suggests that the image is a good candidate for histogram equalization.

```
ImageHistModification root:images:mri
ImageHistogram M_ImageHistEq
NewImage M_ImageHistEq
Display /W=(201.6,45.2,411,223.4) W_ImageHist
```



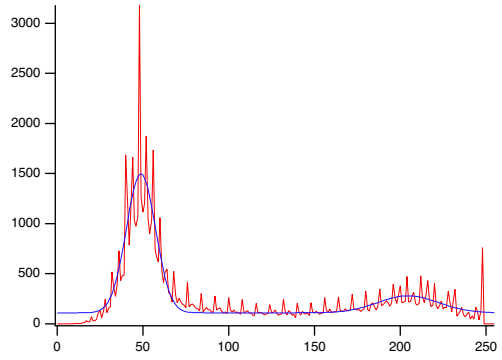
Comparing the two histograms two features stand out: first, there is no change in the dark background because it is only one level (0). Second, the rest of the image which was mostly between the values of 0 and 120 has now been stretched to the range 57-255.

The next example illustrates how you can use the histogram information to determine a threshold value.

```
NewImage root:images:blobs
ImageHistogram root:images:blobs
Display /W=(201.6,45.2,411,223.4) W_ImageHist
```

The resulting histogram is clearly bimodal. Let's fit it to a pair of Gaussians:

```
// Guess coefficient wave based on the histogram.
Make/O/N=6 coeff={0,3000,50,10,500,210,20}
Funcfit/Q twoGaussians,coeff,W_ImageHist /D
ModifyGraph rgb(fit_W_ImageHist)=(0,0,65000)
```



The curve shown in the graph is the functional fit of the sum of two Gaussians. You can now choose, by visual inspection, an x-value between the two Gaussians — probably somewhere in the range of 100-150. In fact, if you test the same image using the built-in thresholding operations that we have discussed above, you will see that the iterated algorithm chooses the value 125, fuzzy entropy chooses 109, etc.

Histograms of RGB or HSL images result in a separate histogram for each color channel:

```
ImageHistogram root:images:peppers
Display W_ImageHistR,W_ImageHistG,W_ImageHistB
ModifyGraph rgb(W_ImageHistG)=(0,65000,0),rgb(W_ImageHistB)=(0,0,65000)
```