# SCALING

## Scaling from Image File Values to the Range of the Display Color Map

### Synopsis

Workstation displays pose a common problem for imaging applications: how to display an image with a large range of values on a screen with a small range of colors. Since there aren't enough colors to use one for each image value, groups of different image values must all be assigned the same color. The process of grouping image values into a smaller set of values is called *scaling* or *binning*.

Use the **Scale** submenu to scale the image to the range of display colors. Any of a variety of linear and non-linear scaling algorithms can be applied simply by clicking on the appropriate submenu button. When scaling is performed, the range of image values currently visible within the area of the display window will be assumed. Limits on the range of image values and other scaling parameters may be input using command line switches. For long integer and floating point data, *SAOimage* uses two stages of scaling, as described below.

### Range of Image Data vs. Size of Color Map

Image data may have a nearly infinite range of data values. Data values may be very large or very small, they may be positive or negative, and they may be real or integer.

A typical workstation can only display 256 colors on its screen at any one time. Some of the 256 colors are needed for coloring such things as the mouse cursor, text characters, background areas, and window borders. On a 256 color workstation, *SAOimage* typically uses 200 colors to display images.

For purposes of display, each pixel in the display image is assigned one of 200 integer values in the range of 0 to 255. Each of these 256 possible values is the index of an entry (or slot) in the display's color map table. The color map table associates specific colors with the 256 display values (e.g. 0=black, 1=dark blue, etc.). (See the *COLOR* section for a detailed discussion of color map manipulation.)

### Basic Scaling Theory

Take the case of an image with a range of values from 0 to +1199 (a range of 1200), and an image display with 200 colors. *SAOimage* might assign one color map entry (or slot) to each range of six in the image. Thus values 0 to 5 would be assigned the lowest color map slot, 0. 6 to 11 would be assigned to 1, and so on up to the values between 1195 and 1199 going to slot 199 in the color map. Then, if slot 0 was black, all pixels in the image with data values between 0 and 5 would appear black in the display. If color map slot 131 was assigned the color pink, then image pixels whose data value was between 787 and 792 would appear pink in the display.

### Scaling Distributions

The above example is an example of linear scaling. But suppose that it was important to be able to see the differences among levels 2, 3, 4, and 5, and not as important to see a difference between 1001 and 1002. Perhaps it would be sufficient to see differences between 1001 and 1100, while 1001, 1002, 1003, etc. could all have the same color. Then it would be better to use a scaling that used more color map slots for the lower image values than the higher image values. Log and square root scaling do that.

To understand how that works, imagine a graph with image values along the side and color map values along the bottom. A plot of linear scaling would be a straight line with a constant slope. In other words, each `n` consecutive image value units would map to `m` color map levels. A log line would be curved, having less slope at the bottom (more color map values per image value range) and get steeper toward the top (fewer color map values per image value range). The square root function would be similar but follow a different curve.

**Non-linear Distributions**

Sometimes weighting toward one end or the other isn't good enough. Perhaps the data values fall in clusters with big gaps in between. Imagine an image with most of its data clustered between 0 and 200, but two or three pixels with a value of 1000. In the linear scaling 80200 and 1000. Worse, as is common with CCD images, there may be bad pixels with values like -1200, which have nothing to do with image source values.

**Histogram Equalization**

A histogram is a plot with the range of possible values (divided into discrete bins) on one axis and the frequency of the actual occurrence of each value, or value within the range of each bin, on the other axis
. The image histogram could be divided into as many bins as colors in the display, with

each bin representing the range of image values associated with a single color in the display. The frequency value of each bin would be the number of pixels in the display that used that color. In a poor scaling distribution, many of the bins (ranges of image values) have few or no actual occurrences in the image.

Histogram equalization is a process of adjusting the ranges of the bins such that each bin has about the same number of image pixels. If the image has many pixels with values between 100 and 200, then histogram equalization would allocate many small bins to that range, (e.g. 100-110, 111-120, etc.). If the image has relatively few pixels with values between 300 and 400, histogram equalization would allocate few large bins to that range (e.g. 300-350, 351-400). The object of histogram equalization is to maximize the information in the display by optimizing the usage of the available colors. It usually produces a dramatic improvement in the amount of visible detail in the displayed image.

The drawback of histogram equalization is that the ranges can vary greatly in size. A difference between two adjacent color map values may represent a big or a small difference in data values, with much irregularity from one step to the next.

**Windowing the Scaling**

Often, most of the data falls within a single contiguous range, with bad pixels or infrequent

events in the extremes to either end. It may be useful to restrict the scaling algorithm to a specific range, with all values below that range having the same color as the minimum and all values above that range having the same color as the maximum. This can either be accomplished by specifying the scaleing limits directly, or by looking for a section of the image which is free from extreme values, and basing the scaling on the values in that section only.

### Wrapping the Color Map

Another trick to show more detail with a limited color map is to reuse it. By wrapping the color map on itself, it could represent the range of values from 0 to 99, and then start over, 100 having the same color as 0 and 199 having the same color as 99. This works for smoothly varying data and even has a sort of contoured look to it. Where the data fluctuations are greater than the range covered by one wrap of the color map, the result is a lot of meaningless noise.

### Realization and Efficiency within SAOimage

In order to realize reasonable execution times for rescaling the image display, *SAOimage* does not apply its scaling functions to real data. Where the image data was real, double, or long, the data is linearly rescaled to integer values between -32767 and 32767 (a range of 65,535). Each scaling function produces a 65,535 entry lookup table for this range of values which is used to draw or redraw the display. If the data has such great extremes that this range linearly applied to the data's range will be insufficient, windowing limits for the initial linear scaling can be given on the command line using `-rmin, -rmax`.

The scaling function is further windowed by being applied only to the range of image values actually being displayed at the time the function is invoked. Pixels not appearing in the display window are not considered in the scaling algorithm. Thus scaling after panning and zooming will produce different scale lookup table mappings, depending on the contents of the display. Panning and zooming can be used in this way to find a better scaling.

To compensate for extreme data that may appear at the edges of an image (common in CCD images) data within 2 pixels of the edge of the display are not considered in the assessment of range. In `verbose` mode, the scaling routine reports the range of values which it finds in the display.

One or both limits of the value range for menu-commanded scaling can be restricted by using the `-min` and `-max` arguments on the command line. When `-min` is given, the low end of the range will not extend below the `-min` value. When `-max` is given, the upper end will not extend above the `-max` value. Limit value are given in terms of original file values. The user need not know about the internally used short integer values. The `-min` and/or `-max` value can be cleared by giving the `-min` or `-max` switch with no argument.

Panning or zooming after a scale map has been made does not cause a new scale map to be calculated. The display is drawn with the existing scaling. If the new display has values outside the range when the map was made, these values are usually clipped (mapped to the color map minimum or maximum). Wrapped scaling is an exception, in that the color map

wrapping is applied all the way to the maximum possible value.

### Linear Scaling

Linear scaling is fairly simple. The range of image values is divided by the number of color map values to determine the range of data values for each color map value. The fixed range is applied to mapping all image values between the minimum and maximum image value. Values below the minimum are all mapped to the lowest color map value. Values above the maximum are all mapped to the highest color map value.

### Wrapped Linear Scaling

Wrapped linear scaling, as described above, divides the range of data values by the number of times the color table will roll over (the default is 10) and the result is divided by the number of color map values to determine the range of data values for each color map value. Values below the minimum are all mapped to the lowest color map value. The mapping continues to be rolled up to the highest map value (32767). (see `-wrap` in the *COMMAND LINE* section)

### Log Scaling

The distribution of data values to color map values is based on the distribution of *e**n* from 0 to *X*, where *n* is a parameter (the default is 10.0) and *X* is determined by *n* and the two ranges (data and color map). Positive values of *n* favor the lower data values, while negative values of *n* favor the higher data values. Values below the minimum are all mapped to the lowest color map value. Values above the maximum are all mapped to the highest color map value. (see `-log` in the *COMMAND LINE* section)

### Square Root Scaling

The distribution of data values to color map values is based on the distribution of *X**(1/n)* from 0 to 1, where *n* is a parameter (the default is 2.0) and *X* varies from 0 to 1 in steps determined by the number of color map values. Values of *n* greater than 1 favor the lower data values, while values of *n* less than 1 favor the higher data values (negative values and 0 are not allowed). Values below the minimum are all mapped to the lowest color map value. Values above the maximum are all mapped to the highest color map value. (see `-sqrt` in the *COMMAND LINE* section)

### Histogram Equalization

The principle of histogram equalization is described above. The histogram equalization algorithm in *SAOimage* differs from common one-pass algorithms by accounting for disproportionately large occurrences of one or a few image values. In an image with 1000 pixels, 500 of which have the value 32, a one-pass algorithm, given 100 colors, will try to get 10 pixels for each color. It will end up with 49 unused colors, since one color covered 500 pixels. *SAOimage's* algorithm detects the peak (or peaks) in the histogram and allocates the remaining 99 colors among the 500 remaining pixels (5 pixels per color).

## IRAF

Images sent directly by *IRAF* are already scaled to the range 1-200. *IRAF's* own scaling defaults to a windowed linear scaling where the window is determined by fitting a straight line to a small, 200 pixel, subsampled histogram. Further rescaling within *SAOimage* is not very useful (see the *IRAF* section).