Subject: Re: Sorting and image rescaling
Posted by John-David T. Smith on Wed, 30 May 2001 16:44:05 GMT
View Forum Message <> Reply to Message

Bill wrote:
>
> I work on some of the software for an imaging sensor. One of the codes I
> wrote was to generate JPEGS. In order to generate prettier JPEGS I wrote
> my own bytescaling routine that crudely takes into account the
> distribution of values within a band image. This routine relies on a
> single line equivalent to
>
> sorted_values = SORT(band_image)
>
> where SORT is the IDL intrinsic, and band image is a floating point two
> dimensional array. I basically use this line to find a set percentage
> of minum and maximum outliers , and use the maximum and minimum of the
> remaining inliers to do a linear rescaling to values from 0 to 255.
>
> Some of our band images are on the order of 2500 by 10000. For such band
> images this line can take over 30 seconds per band. This is a moderate
> nuisance at the moment, but we are planning to update our calibration ,
> and reprocess 1000s of multiband images with a new calibration.
> Naturally we want to update the jpegs to reflect this new calibration.
> It appears that this single line will extend reprocessing by a couple of
> days. I don't like this. This yields the following questions:
>
> 1. Does anyone know a better general approach to such a rescaling that
> avoids the need to sort the data, or sort more than a fraction of the
> data?
>
> 2. How does ENVI do its linear, gaussian, and uniform rescalings? They
> seem to take about a second for these images, so they must be doing
> something different from what I am doing.
>
> 3. Does IDL have a particularly inefficient SORT method for floats? Note
> that for floats it is possible to sort in 0(N), using something like a
> bucketsort, but more flexible sorting routines such as merge sort, heap
> sort, and quick sort are of order 0(N ln(N)).

Did you try RSI's own hist_eq() histogram equalizer? It may do close to what you want, and in any case can serve as a starting point. You construct the cumulative histogram and use it to map the color ramp. Histogram will be faster than sort for well behaved arrays (i.e. not super-sparse), and with a well-chosen binsize. Speed will come with a large binsize. Since you only have 256 final values among which to choose, a miniscule binsize is unecessary. At some level though, working with such large arrays will be slow without lots of memory. One

other possibility is max min clipping: just scale to 5% inside of the bounds, which will work fine for well behaved images (no off-scale pixels).

JD