Subject: Sorting and image rescaling
Posted by William Clodius on Fri, 25 May 2001 22:34:04 GMT
View Forum Message <> Reply to Message

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)).

Subject: Re: Sorting and image rescaling
Posted by Martin Schultz on Mon, 28 May 2001 16:54:38 GMT
View Forum Message <> Reply to Message

Bill <wclodius@lanl.gov> writes:

> 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)).
>
>

This sounds like a prime time application for histogram ;-) Just don't ask me how to use this magical hat ;-(

Cheers,
Martin

--
[[[[[[[[[
[[Dr. Martin Schultz   Max-Planck-Institut fuer Meteorologie   [[
[[Bundesstr. 55, 20146 Hamburg   [[
[[phone: +49 40 41173-308   [[
[[fax: +49 40 41173-298   [[
[[martin.schultz@dkrz.de   [[
[[[[[[[[[

Subject: Re: Sorting and image rescaling
Posted by Craig Markwardt on Mon, 28 May 2001 17:25:16 GMT
View Forum Message <> Reply to Message

Martin Schultz <martin.schultz@dkrz.de> writes:

> Bill <wclodius@lanl.gov> writes:
>
>> 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.
... Clodius text deleted by CM ...
>
> This sounds like a prime time application for histogram ;-) Just
> don't ask me how to use this magical hat ;-(

I agree with Martin here, but you can probably make it as simple or
complicated as you want.

The simplest way to do it would be to construct a histogram and then
use a FOR loop to nibble x% from the edges, where x is selectable. In
newer versions of IDL you can use the CUMULATIVE option to the TOTAL
function to simplify life a little further, since it avoids the FOR
loop.

I don't think the REVERSE_INDICES keyword of HISTOGRAM helps a whole
lot in this case (whew, some people say). The only benefit is that
the REVERSE_INDICES keyword returns some information on the cumulative
distribution which you might use to your advantage (ie, avoid the FOR
loop again). This appears to be overkill though, and has the
potential to use a lot of memory like the SORT solution does.

Craig

--
 -- --------------
Craig B. Markwardt, Ph.D.   EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
 -- --------------

Subject: Re: Sorting and image rescaling
Posted by Wayne Landsman on Mon, 28 May 2001 21:50:45 GMT
View Forum Message <> Reply to Message

Bill wrote:

>
> 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?

Provided your images are well-behaved, I would think it would suffice to sample a few thousand pixels distributed uniformly over your image to establish the scaling. For all the but the most bizarre histograms of pixel intensities, I would think a few thousand samples is a good approximation.

An astronomical application can be found in the program sky.pro in http://idlastro.gsfc.nasa.gov/ftp/pro/idlphot/. This program establishes a grid of about 4000 pixel values uniformly distributed across the image. It then runs a fairly complicated program to throw out outliers (assumed in astronomy to be overwhelmingly positive) to establish a mean sky and sigma. For display purposes one can then choose to display between say sky - 2*sigma to sky + 20*sigma.

--Wayne Landsman landsman@mpb.gsfc.nasa.gov

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