Subject: rebin question
Posted by Jonathan Joseph on Fri, 22 Mar 2002 16:58:41 GMT
View Forum Message <> Reply to Message

I figured I would use rebin to downsample an image by averaging the pixels in blocks of specified size. What I discovered, was that for integer type images, rebin averages the pixels, but then instead of rounding to the nearest integer, simply takes the integer part of the average. Hence:

print, rebin([5,5,5,5,4], 1)

gives the result of 4, not 5 which is what I would like. I suppose this is done for speed - to work around the problem, I need to convert to a floating point type, do the rebin, then round, then convert back to the proper integer type - a hassle.

But, I would really like a more generic way of doing downsampling of this sort, without the high overhead of a loop. Apart from taking the mean of a block of pixels, I would also like the option of downsampling using the median of a block of pixels, or using the mean of a block of pixels disregarding the farthest outlier (or n outliers).

Has anyone written IDL code to do downsampling in a more generalized way than rebin, or have any clever ideas about how to do it quickly?

Thanks

Subject: Re: rebin question
Posted by Wayne Landsman on Fri, 22 Mar 2002 19:41:31 GMT
View Forum Message <> Reply to Message

Jonathan Joseph wrote:

> I figured I would use rebin to downsample an image by averaging the
> pixels in blocks of specified size. What I discovered, was that for
> integer type images, rebin averages the pixels, but then instead of
> rounding to the nearest integer, simply takes the integer part of
> the average. Hence:
>
> print, rebin([5,5,5,5,4], 1)
>
> gives the result of 4, not 5 which is what I would like. I suppose
> this is done for speed - to work around the problem, I need to convert
> to a floating point type, do the rebin, then round, then convert back
> to the proper integer type - a hassle.

You might want to look at the program boxave.pro in
http://idlastro.gsfc.nasa.gov/ftp/pro/image/ which does the general
operation (but note that it wants to the binning size and not the output
image size). It uses loops, but only on the binning indicies, i.e. to
reduce an array by a factor of four requires a loop over 4 numbers.

Also in that directory is my favorite resizing program -- frebin.pro.
I think of it as the best of both REBIN and CONGRID. LIke REBIN it
conserves flux by making sure that every input pixel is equally
represented in the output array. But like CONGRID it allows
redimensioning to arbitary sizes. (FREBIN always returns at least
floating point, so it is not quite what you are looking for).

On a somewhat related topic, I am currently investigating writing a
"flux-conserving" version of POLY_2D, which is used in rot.pro and other
image warping procedures. The linear interpolation mode of POLY_2D
works by examining which input pixels contribute to a given output pixel,
and applying a linear interpolation according to the weight of the
contribution. But this does not guarantee that every input pixel is
equally represented in the output array, and so, for example, using
rot.pro on an image with a (undersampled) star, will not conserve the
flux of the star. (Note that this is has nothing to do with the edge
effect problems with rot.pro discussed in an earlier thread.).

To conserve the flux, you have to reverse the process -- start with an
input pixel, and follow it through the transformation and "drizzle" it
onto the output array, computing the fraction of overlap of the
transformed input pixel with each output pixel. This is a process
that almost certainly cannot be vectorized -- since the transformed
input pixel will usually fall at an angle on the output array, and one
has to compute the overlap with pixels in the output grid. (In the
usual astronomical "drizzle", the size of the pixel is also allowed to
change (decrease) during the transformation.). So I suppose it will
have to be written as a linked C program, though I welcome any ideas.

I'm not sure how generally useful this would be, but astronomers are
fussy about losing any of their precious photons.

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