Subject: Re: rebin question
Posted by Jonathan Joseph on Fri, 22 Mar 2002 18:40:32 GMT
View Forum Message <> Reply to Message

It looks nice doesn't it, and I did write a procedure for the simple
case of averaging, but it's not as clean cut as you indicate:

1. first one needs to get the type of the incoming image - you don't
   want to round the result of a floating point type image - that
   would give you the wrong result.

2. conversion should be done to double precision floating point
   (not float) otherwise large long integers will lose precision.
   loss of precision for large L64 integers will occur even with
   conversion to double, so they can't be handled properly at all.

3. need to convert back to the proper type, so your solution
   should be wrapped by a fix(..., type=type)

4. instead of a rebin, there is now a rebin, two type conversions
   and a round, which will slow things down and use more memory.

So, it is a hassle.

But yes, it's still not difficult to write a function to handle the
SIMPLE case of averaging for CERTAIN data types. But that does not
help with the problem of writing a more general function that handles
downsampling using median or downsampling using a mean excluding
outliers (pixels with values far from the mean) or downsampling using
your favorite method. Doing this quickly in IDL means doing it
w/o loops, so while conceptually the problem is not difficult, it
does seem somewhat more difficult to do it properly in IDL.

Anyone out there thought about this problem before?

-Jonathan

Vince wrote:
>
> print, round(rebin(float([5,5,5,5,4]),1))
>
> Hassle?
>
> Maybe you could write a function. Which leads me to a new question:
>
> Is it possible to define a function or procedure in IDL that can take
> an arbitrary number of arguments, e.g.:
>

> function my_rebin, a, arg1, arg2, ...
>
> return, round(rebin(float(a), arg1, arg2, ...))
> end
>
> On Fri, 22 Mar 2002 11:58:41 -0500, Jonathan Joseph <jj21@cornell.edu>
> 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.
>>
>> 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