Subject: Re: Maximum value array resampling
Posted by JD Smith on Fri, 05 Aug 2005 23:09:55 GMT
View Forum Message <> Reply to Message

On Fri, 05 Aug 2005 10:55:46 -0700, rechoncho@yahoo.com wrote:

```
>  I'm trying to figure out an IDL-efficient way to resample a series of
>  images. I know how to do this is a ruinously laborious fashion using loops
>  but I know there's any easier way.
>
>  Consider the following 4x4 array:
>
>  x = [$
>  [0,3,4,5],$
>  [1,2,7,0],$
>  [3,2,9,0],$
>  [7,0,5,6]]
>
>  I want to resample this to y, a 2x2 array. Each element would contain the
>  maximum value of the corresponding 4 pixels. y would then look like
>
>  [$
>  [3,7],$
>  [7,9]]
>
>  So element [0,0] in y is max(x[0:1,0:1]) etc. As I understand it,
>  rebin/congrid won't do this. Each image is about 5000x2000 and there are
>  several hundred to process.
```

For arbitrary images with both dimensions even:

```
d=size(x,/DIMENSIONS) & nx=d[0]/2 & ny=d[1]/2
 y=transpose(max(reform(transpose(reform(x,2,nx,2*ny),[0,2,1] ), $
                      4,ny,nx),DIMENSION=1))
```

How does it work? It juggles dimensions so that the indices of all the 2x2 sub-arrays are next to each other in memory, and then uses max(/DIMENSION) to collapse over them. The inner call to REFORM puts them adjacent to each other, but in the wrong dimension, then TRANSPOSE makes them adjacent in the fast-changing dimension. The rest is straightforward.

There may be a quicker way with only one call to TRANSPOSE, but I couldn't find it (anyone?). Also, if you don't care to keep X, throw a couple of /OVERWRITE keywords for both REFORM statements, to save some memory and time.

JD