Subject: Re: ENVI_INIT_TILE tiling problem
Posted by jeffnettles4870 on Tue, 17 Mar 2009 01:16:46 GMT
View Forum Message <> Reply to Message

On Mar 16, 4:57 pm, a.l.j.f...@gmail.com wrote:
> On Mar 16, 12:47 pm, "Jean H." <jghas...@DELTHIS.ucalgary.ANDTHIS.ca>
> wrote:
>
>
>
>>> But this now results in the following error, which I don't understand
>>> and can't find a reference to (including in tri_surf.pro, unless I've
>>> missed something?). Any ideas?
>
>>> % Array dimensions must be greater than 0.
>>> % Error occurred at: TRI_SURF 136 C:\Program Files\ITT
>>> \IDL64\lib\tri_surf.pro
>
>>> output_DSM is certainly a 2D array, as I'm able to TVSCL it to view
>>> the contents.
>
>>> output_DSM contains lots of holes (value of 0) which I'd like to
>>> interpolate over using TRI_SURF.
>
>> lines 135 and 136 are:
>> if n_elements(xgrid) eq 2 then begin
>> x = findgen(nx) * xgrid[1] + xgrid[0]
>> NX being computed form the Z input (your output_dsm array)
>
>> so, the problem is your Z input... double check, just before calling
>> tri_surf, what is the content and size of output_dsm!
>
>> Jean
>
> Hi Jean,
>
> Yes, yet again you were absolutely correct. the problem was with my
> "output_dsm". In order to get a FID from output_dsm I used
> ENVI_ENTER_DATA (maybe this isn't the best way??), which then seemed
> to prevent output_dsm being used as an array. Therefore, before I used
> ENVI_ENTER_DATA I made a copy of output_dsm, called output_dsm_copy!
> This meant it was preserved as an array. This might not be the best
> way to do things (?), but it worked.
>
> Things are now working mostly OK and the tiling and interpolation
> appear to complete, except that the zero pixels in my array we're
> interpolated over. Therefore I changed my interpolation to the
> following (I decided to go with MIN_CURVE_SURF in this example, but

> the same should be true for TRI_SURF):
>
> tile_id=ENVI_INIT_TILE(fid_output_DSM, my_pos,
> num_tiles=number_of_tiles)
> FOR i=0, number_of_tiles-1 DO BEGIN
> tile_data_interp=ENVI_GET_TILE(tile_id, i)
>
> ;Processing within Tiling
>
> ;tile_data_interp = REPLICATE(0.0, dims[2], dims[4])
>
> index= WHERE (output_dsm_copy GT 0.0)
>
> x = index MOD DIMS[2]
> y = index/DIMS[4]
>
> z = output_dsm_copy [index]
>
> tile_data_interp = MIN_CURVE_SURF (z, x, y, gs=[1,1],bounds=[1,1,DIMS
> [2],DIMS[4]])
>
> ;tile_data_interp = TRI_SURF(output_DSM, /REGULAR, XGRID=[1, 1],
> YGRID=[1, 1], NX=dims[2], NY=dims[4])
>
> ; Close Tiling
>
> ENDFOR
> ENVI_TILE_DONE, tile_id
>
> Can you spot the problem?? When I run it the interpolation runs out of
> memory for creating the array (% Unable to allocate memory: to make
> array.
> Not enough space). This is because I'm using DIMS for the original
> file outside of the tiling... whereas I need to use different, smaller
> DIMS within the tiles (the x,y, dimensions of the tiles themselves).
> How can I get the tile dimensions and use them here??
>
> Many thanks again!

There's a couple ways you can figure this out. You can actually control the interleave of the tile, which would mean you would know the tile dimensions most of the time. This isn't true 100% of the time, but i've never run into a case where it wasn't. But anyway, a pretty reliable way to do this would be to put this code in your tile loop:

if i eq 0 then s = size(tile_data, /dimensions)

which would make s[0] the size in the x direction and s[1] size in the y direction. You can feed that into min_curve_surf or tri_surf later.