Subject: Re: weird behavior of Triangulate
Posted by Yngvar Larsen on Mon, 10 Sep 2012 09:00:38 GMT
View Forum Message <> Reply to Message

On Thursday, 6 September 2012 18:05:39 UTC+2, Coyote wrote:
> I wrote earlier:
>
>
>
>> It seems as if GridData would be useful. But, unless you are
>
>> working with 50x50 pixel blocks of data, GridData is ungodly
>
>> slow! Interpolate is probably better, but you don't have
>
>> many gridding options (linear, bilinear, cubic).
>
>
>
> By the way, I've demonstrated that GridData *does* do the right
>
> thing (sorta) *if* the data set resolution AND the output grid
>
> resolution is small enough. (And, maybe, if you run it on the
>
> right machine.) But, I need a more robust solution that works
>
> with the kinds of satellite data I use daily.
>

So your problem is basically that GRIDDATA is slow?

From my point of view, GRIDDATA is for gridding _irregular_ data, which is a hard problem. If your data is already on some regular grid, why would you want to triangulate? Regular interpolation is all that is needed if you do it the right way.

In general, I do the following to transform data from one grid to another:

1) Divide the _output_ grid in manageable blocks. What "manageable" means, will depend on many things, but for satellite data mostly on memory limitations. Overlap between the blocks might be necessary if you are going to include filtering/interpolation.

Repeat the following for your each of the output blocks:
2a) Calculate coordinates in _input_ grid corresponding to the grid points in your current output grid.
2b) Convert these coordinates to indices in the input data grid, including subpixels.
2c) Extract from your input dataset a "big enough" tile from the input grid. Or keep the entire input dataset in memory if it is already small enough.

2d) Choose gridding mode. If your output grid resolution is approximately the same or higher than the input grid resolution, nothing is needed here. However, if the output resolution is coarser, you might want to do something to avoid undersampling. E.g., smooth input data to match output resolution. Of course, if you have some missing data in the input grid, you must be careful at this point.
2e) Perform the actual regridding, using mapping indices from (2b), say XIND and YIND.
Nearest neighbour: outdata = indata[round(xind), round(yind)]
Bilinear interpolation: outdata = interpolate(indata, xind, yind)
Cubic interpolation: outdata = interpolate(indata, xind, yind, cubic=-0.5)
Again, take care if you have missing data in your input.

3) Glue together output blocks after elimination of possible block overlap.

Notes:
* For the regridding operation in (2e), cubic interpolation kernels might for some reason still not be good enough for you. In that case, it is not hard to implement fairly fast kernel based interpolators in IDL.

* For some kinds of satellite data with high dynamic range, it is sometimes better to perform the interpolation on the logarithm of the original data.

--
Yngvar
