Subject: Re: four corners for lat & long pixels
Posted by James Kuyper on Tue, 03 Jul 2007 10:40:39 GMT
View Forum Message <> Reply to Message

titan wrote:
> I have a radar image and I know latitude and longitude of each pixels
> of this image
> I wonder how to get latitude and longitude of the four corners
> "surrounding" each pixels of the image

A good approximation would simply assign to each corner a latittude equal to the average of the latitudes of pixels that meet at that corner, and the same for longitudes. For the pixel corners on the outer edge of the image, use bilinear extrapolation from nearby pixels. Note that it takes special handling to make this approach work properly if your image crosses the meridian at 180W = 180E. This approach will unavoidably produce bad results near the north and south poles.

For a more sophistocated analysis, you could use 2-D spline interpolation. However, if you're going to get that sophisticated about it, you should convert your latitudes and longitudes into map coordinates for a map projection that reflects, with reasonable accuracy, the way in which your image was collected. An ideal map projection for your particular image would convert the latitudes and longitudes of each row or column of your image into a straight line of evenly spaced dots. Interpolate/extrapolate the projected coordinates, just as described above for the latitude and longitude, to get projection coordinates for pixel corners. Then invert the map projection to get latitude and longitude for those corners. In addition to getting slightly better accuracy, interpolation with a well-chosen map projection will also automatically avoid the problems at 180W=180E, and at the north and south poles, without requiring any special handling in your code - the special handling is hidden inside the map projection routines.

Subject: Re: four corners for lat & long pixels
Posted by titan on Thu, 05 Jul 2007 06:44:09 GMT
View Forum Message <> Reply to Message

thank you for your answer but unfortunately i'm trying this procedure for the first time!! :o(
could you please be more clear?

thank you very much

Subject: Re: four corners for lat & long pixels
Posted by James Kuyper on Thu, 05 Jul 2007 16:26:31 GMT
View Forum Message <> Reply to Message

titan wrote:
> thank you for your answer but unfortunately i'm trying this procedure
> for the first time!! :o(
> could you please be more clear?
>
> thank you very much

Let's assume that 'latitude' is an array with dimensions (cols, rows). Most of the work that needs to be done can be done with the following commands:

```
ix = DINDGEN(cols+1)-0.5
iy = DINDGEN(rows+1)-0.5
corner_lat = BILINEAR(latitude, ix, iy)
```

Note: ix and iy have now been replaced by two-dimensional arrays. Unfortunately, when it goes outside of the bounds of the input array, BILINEAR switches over to a nearest-neighbor rather than using bilinear extrapolation. IDL has many routines for performing various kinds of interpolation, but a quick survey didn't turn up a single one that switched over to extrapolation beyond the boundaries of the original data. Therefore, you'll have to fix up all four edges. I'll show you how to handle the first row; the technique for the other three edges are similar:

```
top = latitude[0:cols-2,0:1]+latitude[1:*,0:1]
corner_lat[1:cols-1,0] = (2.0*top[*,0] - top[*,1])/6.0
```

Finally, special handling is also needed for the four outermost corners. For the first row and column, bilinear extrapolation to the corner point is equivalent to:

```
corner_lat[0,0] = 0.5625*latitude[0,0]-.
1875*(latitude[1,0]+latitude[0,1])+0.1875*latitude[1,1]
```

Similar calculations apply at the other three corners.

You shouldn't worry about the more sophisticated approaches I mentioned earlier, until you understand this simple approach well enough.