Subject: Re: Convert planetary sinusoidal map image to (lat,lon,value) array
Posted by JD Smith on Wed, 31 Jul 2002 01:15:36 GMT
View Forum Message <> Reply to Message

On Mon, 29 Jul 2002 11:27:08 -0700, David Fanning wrote:

```
> Ken Mankoff (mankoff@I.HATE.SPAM.cs.colorado.edu) writes:
>
>> I guess I'll give this a shot. Never done this but here is what I would
>> try first if I were doing it:
>
> Not too bad, not too bad. :-)
>
> It would certainly work something like this, *if* you could get an IDL
> map projection to work. I have my doubts about what a "sinusoidal equal
> area" projection is, but assuming it is the same thing as a sinusoidal
> projection in IDL, I would make a few modifications to your suggestions.
>
>> 1) Set up an identical map projection. The key word (no pun intended)
>> here is *identical*, so that every pixel of your new "sinusoidal equal
>> area"  projection is the same as the BYTARR you have been given. If you
>> can't get it identical, I am not sure what to do next.
>>
>> You can test if it is identical via:
>> IDL> WINDOW, XSIZE=1440, YSIZE=720
>> IDL> MAP_SET, 0, 0, /SIN, /ISO
>> IDL> TV, sinusoidal_projection
>> IDL> MAP_GRID & MAP_CONTINENTS & MAP_HORIZON
>>
>> Does the border created by MAP_HORIZON line up exactly (to the pixel)
>> with the data your data from the TV command? Look up the keywords to
>> MAP_SET if not, and try other projections...
>
> I would add MARGIN=0 and NOBORDER=1 keywords to the MAP_SET command. And
> if I knew (or could figure out) the lat/lon coordinates of the corners
> of the image I would add the 8-element version of the LIMIT keyword as
> well. Then there is a reasonably good chance the map might match the
> image.
>
> You could set up the map coordinates in a pixmap if you didn't want to
> see something happening on the display. Just make the pixmap the same
> size as your image, etc.
>
>> 2) Redo the above code, but stop after the TV command so its just your
>> data, nothing extra. You don't actually need a window, but you need the
>> MAP_SET command to be run (with the correct 1440,720 sizes) so that IDL
>> defines the map coordinate system. You are going to use this later to
>> convert between (x,y) pixels and (lat,lon) degrees.
```

```
>>
>>  3) Set up the new array you want to put your data into. I suggest a
>>  cylindrical 'projection', which can then be warped to any other
>>  projection you want. So...
>>  IDL> new_array = BYTARR( 1440, 720 )
>
>  I would just leave it in its current projection, assuming this is the
>  correct one.
>
>>  4) Step through every (x,y) pixel in your image.
>
>  Since the original poster indicated that he was only interested in
>  non-zero values, I would just work with those.
>
>>  FOR x=0,1439 DO BEGIN
>>      FOR y=0,719 DO BEGIN
>>   aPixel = sinusoidal( x, y )
>>   IF ( aPixel NE 0 ) THEN BEGIN ; not a valid (lat,lon) coord.
>>       latlon = CONVERT_COORD( x, y, /device, /to_data ) new_array[
>>       latlon[0], latlon[1] ] = aPixel
>>          ENDIF
>>      ENDFOR
>>  ENDFOR
>
>  I would do it something like this:
>
>    indices = Where(sinusoidal_projection GT 0, count) IF count EQ 0 THEN
>    Message, 'Whoops. Somethin gone wrong!!' x = indices MOD 1439 y =
>    indices / 1439
```

I think you meant 1440.

```
>    latlon = CONVERT_COORD( x, y, /DEVICE, /TO_DATA )
>
>  Now, the information you want is in latlon.
```

Not to oversimplify, but the sinusoidal projection is given by:

```
 x = longitude*cos(latitude)
 y = latitude
```

You just need the inverse of this simple transform.

The y transform is trivial: knowing the range of latitude present in your grid (lat_high and lat_low), you can write:

```
IDL> lat=findgen(720)/719*(lat_high-lat_low)+lat_low
```

OK, now you have the latitude for each row in your image. Given the known range of longitudes present in your grid (lon_low to lon_high), you can calculate:

```
IDL>  lon=(findgen(1440,720) mod 1440/1439*(lon_high-lon_low)+lon_low)/ $
          rebin(reform(cos(lat),1,720),1440,720)
```

Here I've simply linearly mapped the full range of x to the full range of longitude (which is only valid at the equator), then divided by a suitably inflated cos(lat). This will of course contain spurious values for the longitude in the corners, which you can clip with:

```
IDL>  lon[where(lon gt lon_high OR lon lt lon_low)]=!VALUES.F_NAN
```

And you might convince yourself it's right with:

```
IDL> WINDOW, XSIZE=1440, YSIZE=720
IDL> tmp=lon & tmp[where(finite(tmp) eq 0b)]=0. & tvscl,tmp
```

Try this with:

```
IDL>  lat_low=-!PI/2 & lat_high=!PI/2 & lon_low=-!PI & lon_high=!PI
```

and you'll see a lovely map of longitude over the full globe. This also reveals that the other suggested method (convert_coord) fails, without the all-important POSITION keyword:

```
IDL>   map_set,0,0,/SIN,/ISO,/NOBORDER,/NOERASE,POSITION=[0.,0.,1., 1.]
IDL>  map_continents
IDL>  map_horizon
IDL>  map_grid,LONDEL=15,LATDEL=1
```

Remember that near the poles, cos=0, and the longitude is singular. What this means in practice is that in these regions the longitude mapping is much less certain, and subject to a greater degree of roundoff-induced error, etc. The lat-lon bins are also severely distorted.

Good luck,

JD