
Subject: Re: ms2gt MODIS reprojection toolkit
Posted by [Klemen](#) on Wed, 28 Apr 2010 15:34:37 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi Maarten,

try to simply interpolate the values in IDL to a sphere. SPH_SCAT does exactly what you want but its output might be strange if you don't have the global coverage.

I wrote an example of using TRIANGULATE + GRIDDATA. You can rewrite it so it can fit your requirements, you probably need to calibrate the data, etc.

Cheers, Klemen

```
; modis_swath2_grid.pro
; converts swath to a regular geographical grid named
; "Equatorial Cylindrical Equidistant", "Plate-Caree", "simple
; cylindrical", "lat/lon", or sometimes "unprojected"
; works for MOD021KM and MYD021KM

; in_file - input level 1B MODIS data
; dataset - "EV_1KM_Emissive", or "EV_250_Aggr1km_RefSB", or
; "EV_500_Aggr1km_RefSB"
; layer_num - the corresponding number of layer within the dataset,
; e.g. 0 for band 1 at 0.6 microm, or 10 for band 31 at 11 microm
; d_resolution - spatial resolution of output (in this case decimal
; degrees)

; Run as:
; a =
; modis_swath2_grid('MYD021KM.A2010107.1000.005.2010109145717. hdf',
; 'EV_1KM_Emissive', 10, 0.1)

; klemen.zaksek@zmaw.de, 2010
```

Function modis_swath2_grid, in_file, dataset, layer_num, d_resolution

```
;Tie points
d_xL = -180. ;left
d_xR = 180. ;right
d_yA = 90. ;above
d_yB = -90. ;below

; Set GeoTiff geotags: http://www.remotesensing.org/geotiff/spec/contents.html
```

```

s_geotag = {$
  MODELPIXELSCALETAG: [d_resolution, d_resolution, 0], $ ;resolution
  MODELTIEPOINTTAG: [ 0, 0, 0, d_xL, d_yA, 0], $ ;coordinates left
above
  GTMODELTYPEGEOKEY: 2, $ ;Geographic latitude-longitude System
    GTRASTERTYPEGEOKEY: 1, $ ;raster type
  GEOGRAPHICTYPEGEOKEY: 4326, $ ;geodetic datum WGS84
  GeogPrimeMeridianGeoKey: d_xL, $ ;prime
meridian
  GEOANGULARUNITSGEOKEY: 9102} ;angular unit decimal degree

;#####
#####

;Open selected HDF file, read the chosen dataset, and extract the
chosen band
i_fid = EOS_SW_OPEN(in_file, /READ) ;open file
if i_fid eq -1 then begin
  print, 'The input file does not exist or is not EOS HDF format!'
  GOTO, JUMP1
endif
i_NSwath = EOS_SW_INQSWATH(in_file, s_SwathList)
i_swathID = EOS_SW_ATTACH(i_fid, s_SwathList) ;attach object
i_status_read = EOS_SW_READFIELD(i_swathID, dataset, m_modis) ;read
1000m data
i_status_read = EOS_SW_READFIELD(i_swathID, "Latitude", m_lat) ;read
5000m latitude
i_status_read = EOS_SW_READFIELD(i_swathID, "Longitude", m_lon) ;read
5000m longitude
i_status_detach = EOS_SW_DETACH(i_swathID) ;detach object
i_status_close = EOS_SW_CLOSE(i_fid) ;close file
m_modis = m_modis[*,*,layer_num] ;extract the chosen band
in_size = size(m_modis)
x_min = floor(min(m_lon) / d_resolution) *
d_resolution ;interpolation borders
x_max = ceil(max(m_lon) / d_resolution) * d_resolution
y_min = floor(min(m_lat) / d_resolution) * d_resolution
if y_min eq -90. then y_min = y_min + d_resolution ;life is easier
if you do not consider data on the poles
y_max = ceil(max(m_lat) / d_resolution) * d_resolution
if y_max eq 90. then y_max = y_min - d_resolution
x_pix_min = (x_min - d_xL) / d_resolution
x_pix_max = (x_max - d_xL) / d_resolution
y_pix_min = (d_yA - y_max) / d_resolution
y_pix_max = (d_yA - y_min) / d_resolution

;#####
#####

```

```

; Average original data to "geolocation frame"
;first prepare indexes
out_size = size(m_lat) ;the output will have a reduced spatial
resolution (corresponding to the geolocation)
;the position 0,0 in geolocation corresponds to pixel 2,2 in original
data
;the geolocation is 5 times downsampled
out_indx_col = indgen(out_size[1]) * 5L + 2L ;corresponding coloumns
of orig. data in downsampled grid
out_indx_lin = indgen(out_size[2]) * 5L + 2L ;corresponding lines of
orig. data in downsampled grid
out_indx_col = rebin(out_indx_col, out_size[1], out_size[2])
out_indx_lin = rebin(reform(out_indx_lin,1,out_size[2]), out_size[1],
out_size[2])
out_indx = out_indx_lin * in_size[1] + out_indx_col ;one dimensional
index of original data in downsampled grid

;compute mean value for the radiance
m_count = make_array(out_size[1],out_size[2]) ;array containing the
number of good maeasurements
m_mean = make_array(out_size[1],out_size[2]) ;array containing the
mean maeasurements
for j=-2,2 do begin
  for i=-2,2 do begin
    indx = out_indx + out_size[1]*j + i
    tmp = m_modis[out_indx]
    indx_good = where(tmp le 32767) ;do not use nodata, etc.
    m_count[indx_good] = m_count[indx_good] + 1
    m_mean[indx_good] = m_mean[indx_good] + tmp[indx_good]
  endfor
endfor
m_mean = m_mean / m_count

#####

; Interpolate the data to the regular grid
v_x = transpose(m_lon[*]) ;transform into vector
v_y = transpose(m_lat[*])
; m_img0 = SPH_SCAT(v_x, v_y, m_mean, BOUNDS=[x_min, y_min, x_max,
y_max], GS=[d_resolution,d_resolution])
TRIANGULATE, v_x, v_y, trg, SPHERE=sphere, /DEGREES, FVALUE=m_mean
m_img0 = GRIDDATA(v_x, v_y, m_mean, /SPHERE, /DEGREES, /
INVERSE_DISTANCE, TRIANGLES = trg, $
  START = [x_min, y_min], DELTA = [d_resolution,d_resolution], $
  DIMENSION = [x_pix_max-x_pix_min+1, y_pix_max-y_pix_min+1], $
  MISSING=0, MAX_PER_SECTOR=1, SEARCH_ELLIPSE=3*d_resolution)

```

```
;write geotiff
m_img0 = reverse(m_img0, 2)
m_img = make_array((d_xR-d_xL)/d_resolution, (d_yA-d_yB)/
d_resolution)
m_img[x_pix_min:x_pix_max,y_pix_min:y_pix_max] = m_img0
write_tiff, in_file+'.tif', m_img, compression=1, geotiff=s_geotag, /
long
```

```
JUMP1: print, 'END'
return, m_img
```

END
