
Subject: Help Doing HDF(-EOS) to Multi-layer GeoTIFF Conversion with IDL
Posted by [dlamarhawkins](#) on Tue, 02 Mar 2010 17:19:01 GMT

[View Forum Message](#) <> [Reply to Message](#)

I have been tasked to convert HDF and HDF-EOS files to multi-layer GeoTIFF images. My source files are SeaWIFS (HDF) and MODIS L1B (HDF-EOS). HEG might be of some help except that it only runs on 32-bit Linux and we have 64-bit machines. We do, however, have a copy of IDL 6.2 on our Linux machines. Although I've done quite a bit of file conversion in the past, I am completely new to geo-located data.

I am aware of IDL's `HDF_*` and `EOS_SW_*` routines and IDL's `WRITE_TIFF,...GEOTIFF=struct` routine. I just don't know how to translate, for example, MODIS lat/lon data into the information needed in the IDL geotiff structure. Surely someone else here has already solved this problem. Does anyone have example IDL code that performs the necessary translation? Many thanks!

Subject: Re: Help Doing HDF(-EOS) to Multi-layer GeoTIFF Conversion with IDL
Posted by [Li Xu](#) on Fri, 31 Jul 2015 02:48:24 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Wednesday, March 3, 2010 at 1:19:01 AM UTC+8, dlarahawkins wrote:

> I have been tasked to convert HDF and HDF-EOS files to multi-layer
> GeoTIFF images. My source files are SeaWIFS (HDF) and MODIS L1B (HDF-
> EOS). HEG might be of some help except that it only runs on 32-bit
> Linux and we have 64-bit machines. We do, however, have a copy of IDL
> 6.2 on our Linux machines. Although I've done quite a bit of file
> conversion in the past, I am completely new to geo-located data.
>
> I am aware of IDL's `HDF_*` and `EOS_SW_*` routines and IDL's
> `WRITE_TIFF,...GEOTIFF=struct` routine. I just don't know how to
> translate, for example, MODIS lat/lon data into the information needed
> in the IDL geotiff structure. Surely someone else here has already
> solved this problem. Does anyone have example IDL code that performs
> the necessary translation? Many thanks!

Any one know that above? The problem also I countered.

Subject: Re: Help Doing HDF(-EOS) to Multi-layer GeoTIFF Conversion with IDL
Posted by [Klemen](#) on Fri, 31 Jul 2015 14:40:50 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi,
I have a code that reads MODIS data and writes TIFF of the selected channels. Below is just the

interesting part for you, my code includes volcanic hotspot detection... I have deleted that part, I haven't tested it and it might be that something is still missing, but perhaps it helps. However, I am leaving for holidays so I cannot provide any support at all.
Good luck, Klemen

```
; Global data  
PRO_Global_data
```

```
; Geolocation  
d_lon = -71.93      ;approximate longitude of the centre point in decimal degrees  
d_lat = -39.42      ;approximate latitude of the centre point in decimal degrees  
d_resolution = 0.005 ;resolution of the output in decimal degrees  
d_volc_radius = 0.75 ;how far from the centre point (volcano) the GeoTiff should cover  
d_pos_trshld = d_volc_radius + 0.05 ;position threshold around the centre point in degrees  
d_tiff_size = 2.*d_volc_radius/d_resolution + 1. ;size of GeoTiff in pixels  
  
; ; Estimate tie point left above in geographic coordinates  
; m_point_prj = MAP_PROJ_FORWARD(d_lon, d_lat, MAP_STRUCTURE=s_prj_utm) ;project  
d_xL = Round(d_lon / 0.01) * 0.01 - d_volc_radius  
d_yA = Round(d_lat / 0.01) * 0.01 + d_volc_radius  
  
; ; Estimate border points right below for metadata  
d_xR = d_xL + 2.*d_volc_radius  
d_yB = d_yA - 2.*d_volc_radius  
  
;To remove Bow Tie effect in MODIS data, project them to UTM, 250 m resolution  
;but just for the vicinity of the volcano (e.g. radius 15 km)  
; Set projection properties from geolocation (UTM on WGS84)  
i_utm_z = Round((d_lon+183) / 6) ;determine the UTM zone  
i_utm_m = Double(Round((d_lon+3) / 6) * 6) - 3 ;determine the mean meridian  
s_prj_utm = Map_proj_init(101, DATUM=8, /GCTP, CENTER_LONGITUDE=i_utm_m,  
CENTER_LATITUDE=0)  
  
; Estimate tie point left above in the UTM projection (cca 5km)  
i_tiff_dist = 150. ;how many UTM pixels away from center to work  
i_utm_resolution = 100.  
m_point_prj = Map_proj_forward(d_lon, d_lat, MAP_STRUCTURE=s_prj_utm) ;project  
d_xLutm = Round(m_point_prj[0] / 1000.) * 1000. - i_tiff_dist * i_utm_resolution ;round volcano  
coordinates to 1 km  
d_yAutm = Round(m_point_prj[1] / 1000.) * 1000. + i_tiff_dist * i_utm_resolution  
  
; Estimate border points in geographic coordinates for metadata  
d_xRutm = d_xLutm + i_tiff_dist * 2 * i_utm_resolution  
d_yButm = d_yAutm - i_tiff_dist * 2 * i_utm_resolution
```

```

; Set GeoTiff geotags: http://www.remotesensing.org/geotiff/spec/contents.html
tmp1 = 32600 + i_utm_z
tmp2 = 16000 + i_utm_z
IF d_lat LT 0. THEN BEGIN
    tmp1 = tmp1 + 100
    tmp2 = tmp2 + 100
ENDIF
s_geotag = {$
    MODELPIXELSCALETAG: [i_utm_resolution, i_utm_resolution, 0], $ ;resolution
    MODELTIEPOINTTAG: [ 0, 0, 0, d_xLutm, d_yAutm, 0], $ ;coordinates left above
    GEOGGEODETICDATUMGEOKEY: 6326, $ ;geodetic datum WGS84
    GTMODELTYPEGEOKEY: 1, $ ;projected coordinate system
    GTRASTERTYPEGEOKEY: 1, $ ;raster type
    ; GEOGRAPHICTYPEGEOKEY: 4326, $
    GEOGLINEARUNITSGEOKEY: 9001, $ ;linear unit meter
    GEOANGULARUNITSGEOKEY: 9102, $ ;angular unit decimal degree
    PROJECTEDCSTYPEGEOKEY: tmp1, $ ;projected coordinate system UTM; 326zz north,
327zz south
    PROJECTIONGEOKEY: tmp2, $ ;projection UTM; 160zz north, 161zz south
    PROJNATORIGINLONGGEOKEY: i_utm_m, $ ;projection original longitude
    PROJNATORIGINLATGEOKEY: 0.d, $ ;projection original latitude
    ;PROJCOORDTRANSGEOKEY: 11, $
    PROJLINEARUNITSGEOKEY: 9001, $ ;linear unit meter
    PROJFALSEEASTINGGEOKEY: 500000.d, $ ;false easting
    PROJFALSENORTHINGGEOKEY: 0.d} ;false northing

; ; Tiff size
; i_tiff_size = round(i_tiff_dist * 2000. / i_resolution)

;terrain
m_slp = Read_tiff('O:\Etna_topo_corr\DEM\dem_100M_slope_rad.tif')
m_asp = Read_tiff('O:\Etna_topo_corr\DEM\dem_100M_aspect_rad.tif')
terrain_x = Rebin(Findgen(375)*100. + 482500., 375,375)
terrain_y = Rebin(Reform(-Findgen(375)*100. + 4192500.,1,375), 375,375)

; Write global variables to the file
Save, /VARIABLES, FILENAME = 'data.sav'

END

```

```

; Read MODIS level 1b

; Input parameters are read from ASCII file 'tmp_modis.txt':
; MODIS geolocation file (HDF, 5-min swath, only 1000m data)
; MODIS level1b file (HDF, 5-min swath, only 1000m data)

; Predifined inputs in the DATA.SAV:
; longitude of the centre point
; latitude of the centre point
; resolution of the output
; radius from the centre point in kilometers

; Output:
; in the case of detected hotspot, hotspots characteristics are written into the file
; and 2 geotiffs are made for vizualization

; klemen.zaksek@zmaw.de, 2009

```

PRO Read_modisl1b_v2014, in_list, result=v_results

```

;Read input geo- and scientific-filenames from an ASCII file
;ms_list = list_dates('F:\Exupery\data_results\tmp_modis.txt') ;read list with filenames
;ms_list = list_dates('C:\_data\MODIS\Etna200210\tmp_modis.txt')
ms_list = List_dates(in_list)
in_filegeo = ms_list[0] ;in case of more files for the same dat-time,
in_file = ms_list[-1] ;use the first from the list for the geolocation and the last for data
; FILE_DELETE, in_list, /ALLOW_NONEXISTENT

; Wavelengths
d_l_ch2 = 0.8585
d_l_ch6 = 1.640
d_l_ch21 = 3.959
d_l_ch22 = 3.959
d_l_ch29 = 8.550
d_l_ch31 = 11.030
d_l_ch32 = 12.020

; Nominal nadir resolution in km
d_resolution_km = 1.

; Restore global data
_Global_data
Restore, 'data.sav'

; Open EOS-HDF
i_fid = Eos_sw_open(in_file, /READ) ;open file
i_fidgeo = Eos_sw_open(in_filegeo, /READ)

```



```

    m_ch6_geo = m3_500[i_switch_on_col:i_switch_off_col,i_switch_on_lin:i_switch_off_lin,3]
;channel 6; 1.6 micrometers
    m_ch21_geo = m3_1000e[i_switch_on_col:i_switch_off_col,i_switch_on_lin:i_switch_off_lin,1]
;channel 21; 3.9; saturates at 500K
    m_ch22_geo = m3_1000e[i_switch_on_col:i_switch_off_col,i_switch_on_lin:i_switch_off_lin,2]
;channel 22; 3.9; saturates at 335K
    m_ch29_geo = m3_1000e[i_switch_on_col:i_switch_off_col,i_switch_on_lin:i_switch_off_lin,8]
;channel 29; 8.6 micrometers
    m_ch31_geo = m3_1000e[i_switch_on_col:i_switch_off_col,i_switch_on_lin:i_switch_off_lin,10]
;channel 31; 11 micrometers
    m_ch32_geo = m3_1000e[i_switch_on_col:i_switch_off_col,i_switch_on_lin:i_switch_off_lin,11]
;channel 32; 12 micrometers
    m_lon = m_lon[i_switch_on_col:i_switch_off_col,i_switch_on_lin:i_switch_off_lin]
;longitude
    m_lat = m_lat[i_switch_on_col:i_switch_off_col,i_switch_on_lin:i_switch_off_lin]
;latitude
    m_sol_z = m_sol_z[i_switch_on_col:i_switch_off_col,i_switch_on_lin:i_switch_off_lin]*0.01!/RADEG;solar zenith angle
    m_sat_z = m_sat_z[i_switch_on_col:i_switch_off_col,i_switch_on_lin:i_switch_off_lin]*0.01!/RADEG;satellite zenith angle
    m_sol_a = m_sol_a[i_switch_on_col:i_switch_off_col,i_switch_on_lin:i_switch_off_lin]*0.01!/RADEG;solar azimuth angle
    m_sat_a = m_sat_a[i_switch_on_col:i_switch_off_col,i_switch_on_lin:i_switch_off_lin]*0.01!/RADEG;satellite azimuth angle
;SAVE, m_lon, m_lat, FILENAME = 'data_geo.sav'

;mask nodata value and convert to radiances and check if ch22 is saturated (then replace
saturated pixels with ch21)
    m_mask_geo = m_ch32_geo LT 65000 ;bad values are interpolated, they still might exist
on edges
    i_size = i_switch_off_col - i_switch_on_col
    m_ch1_geo = Compute_modis_radiance(in_file, m_ch1_geo, 6, 0)
    m_ch2_geo = Compute_modis_radiance(in_file, m_ch2_geo, 6, 1)
    m_ch6_geo = Compute_modis_radiance(in_file, m_ch6_geo, 9, 3)
    m_ch21_geo = Compute_modis_radiance(in_file, m_ch21_geo, 4, 1)
;in case of saturated pixels in ch22 replace them with ch21
; restore, 'saturated_index.sav'
; m_index21 = m_tmp_ind
    m_ch22_geo = Compute_modis_radiance(in_file, m_ch22_geo, 4, 2)
    Restore, 'saturated_index.sav'
    m_saturated = Make_array(i_switch_off_col-i_switch_on_col+1,
i_switch_off_lin-i_switch_on_lin+1, /long)
    IF i_count_saturated GT 0 THEN BEGIN
        m_ch22_geo[m_tmp_ind] = m_ch21_geo[m_tmp_ind]
        m_saturated[m_tmp_ind] = 1
    ENDIF
    m_ch29_geo = Compute_modis_radiance(in_file, m_ch29_geo, 4, 8)
    m_ch31_geo = Compute_modis_radiance(in_file, m_ch31_geo, 4, 10)

```



```

Triangulate, point_prj[0,*], point_prj[1,*], Trng, TOLERANCE=0.0001
;longitude (not exact, but a good approximation)
m_Xutm = Griddata(point_prj[0,*], point_prj[1,*], m_lon[*], $
  /NATURAL_NEIGHBOR, TRIANGLES=Trng, DELTA=[[i_utm_resolution],[i_utm_resolution]], $
  DIMENSION=[[i_tiff_dist*2.+1.],[i_tiff_dist*2.+1.]], START=[[d_xLutm],[d_yButm]])
m_Xutm = Reverse(m_Xutm, 2)
;latitude (not exact, but a good approximation)
m_Yutm = Griddata(point_prj[0,*], point_prj[1,*], m_lat[*], $
  /NATURAL_NEIGHBOR, TRIANGLES=Trng, DELTA=[[i_utm_resolution],[i_utm_resolution]], $
  DIMENSION=[[i_tiff_dist*2.+1.],[i_tiff_dist*2.+1.]], START=[[d_xLutm],[d_yButm]])
m_Yutm = Reverse(m_Yutm, 2)
;ch22
m_ch22_geo = Griddata(point_prj[0,*], point_prj[1,*], m_ch22_geo[*], $
  /NEAREST_NEIGHBOR, TRIANGLES=Trng, DELTA=[[i_utm_resolution],[i_utm_resolution]], $
  DIMENSION=[[i_tiff_dist*2.+1.],[i_tiff_dist*2.+1.]], START=[[d_xLutm],[d_yButm]])
m_ch22_geo = Reverse(m_ch22_geo, 2)
;ch31
m_ch31_geo = Griddata(point_prj[0,*], point_prj[1,*], m_ch31_geo[*], $
  /NEAREST_NEIGHBOR, TRIANGLES=Trng, DELTA=[[i_utm_resolution],[i_utm_resolution]], $
  DIMENSION=[[i_tiff_dist*2.+1.],[i_tiff_dist*2.+1.]], START=[[d_xLutm],[d_yButm]])
m_ch31_geo = Reverse(m_ch31_geo, 2)
;ch32
m_ch32_geo = Griddata(point_prj[0,*], point_prj[1,*], m_ch32_geo[*], $
  /NEAREST_NEIGHBOR, TRIANGLES=Trng, DELTA=[[i_utm_resolution],[i_utm_resolution]], $
  DIMENSION=[[i_tiff_dist*2.+1.],[i_tiff_dist*2.+1.]], START=[[d_xLutm],[d_yButm]])
m_ch32_geo = Reverse(m_ch32_geo, 2)
;Below watch out -the geotags in geotiff are still made for the "geographic lon lat" projection and
not for UTM
write_tiff, in_file+'_b39.tif', m_ch22_geo, /float, compression=1, geotiff=s_geotag
write_tiff, in_file+'_b11.tif', m_ch31_geo, /float, compression=1, geotiff=s_geotag
write_tiff, in_file+'_b12.tif', m_ch32_geo, /float, compression=1, geotiff=s_geotag

```

END
