
Subject: Re: Satellite image remapping
Posted by [Sean Raffuse](#) on Tue, 03 Jul 2001 19:07:18 GMT
[View Forum Message](#) <> [Reply to Message](#)

Thanks! I knew I wasn't the first.

-Sean

Paul van Delst <paul.vandelst@noaa.gov> wrote in message
news:3B420A4B.684122C8@noaa.gov...

> Sean Raffuse wrote:

>>

>> Hello, I just found out about this newsgroup. I think it will be a
godsend.

>>

>> Anyway, I'm just getting started with IDL. I have done a few things in
the

>> past few months and am working up to a major project. I know there is
an

>> extensive amount of code out there so maybe my problem is already
solved.

>>

>> What I am doing is remapping satellite images, specifically full-disk
images

>> from a GOES satellite. These images are not internally georeferenced.

>> However, there is available a navigation file which gives the lat/lon
for

>> each pixel. What I would like to do is remap this image onto a flat

>> projection. I know I can do it with a brute force method, but I have
the

>> sneaky suspicion that this sort of thing has been done before. Does
anyone

>> have any experience in this arena? I would be thrilled.

>

> Here is something I slapped together one afternoon back in '97 after some
bods at JPL

> asked me for GOES images. It uses Liam Gumley's IMAGEMAP routine

> (<http://cimss.ssec.wisc.edu/~gumley/imagemap.html>) and some of his color
handling routines

> (<http://cimss.ssec.wisc.edu/~gumley/colortools.html>). I think Liam may
have changed the

> calling sequence to IMAGEMAP since I used it back then so after you get
his new version,

> check the call in the code below.

>

> My code looks like a dogs dinner but it got the job done (back when you
could create gifs).

> The single argument "descriptor" was a file that contained the following:

```

>
> 9631000Z
> 9631003Z
> 9631006Z
> 9631009Z
> 9631012Z
> 9631015Z
> 9631018Z
> 9631021Z
> 9631100Z
> 9631103Z
> 9631106Z
> 9631109Z
> 9631112Z
> 9631115Z
>
> Each time tag let me construct filenames like
> 9631000Z.HDR,9631000Z.LON,9631000Z.LAT,9631000Z.001 which contained all
the gory info
> (they were "flat file" outputs from McIDAS area files (don't ask.)
>
> Anyway, most of remap.pro is reading data files. What you want is
something like right at
> the end:
>
>     range = [ min( data ), max( data ) ]
>     erase, 7
>     pos = [ .05, .05, .9, .95 ]
>     map_set, limit = [ 25, -125, 50, -60 ], pos = pos, /noerase
>     title = sat_inst_id + ' Band ' + string( this_band, format =
'(i2)' ) + $
>             ' ' + parameter[ j ] + ' ; ' + date + ' @ ' + time
>     xyouts, 0.5, 0.975, title, /normal, align = 0.5, color = 0,
charsize = 1.5
> **->   imagemap, data, lat, lon, range = range, /noerase, ncolors =
ncolors, bottom =
> bottom, $
> **->       missing = 7B
>     map_continents, / hires, color = 1
>     map_continents, / hires, /usa, color = 1
>     map_grid, label = 1, color = 2
>     plots, [ pos(0), pos(2), pos(2), pos(0), pos(0) ], [ pos(1),
pos(1), pos(3),
> pos(3), pos(1) ], $
>         /normal, color = 0
>     colorbar, pos = [ .935, .05, .96, .95], title = parameter[ j ],
format = '(f5.1)',
> $

```

```

>         /right, /vertical, min = range[ 0 ], max = range[ 1 ],
divisions = 8, $
>         ncolors = ncolors, bottom = bottom, color = 0
>
> which does all the pretty remapping, etc.
>
> Liam's IMAGEMAP routine is a beaut - sure as hell made my life much
simpler.
>
> paulv
>
>
>
> <-----cut here----->
>
>
> pro remap, descriptor, ps = ps, gif = gif, no_pause = no_pause
>
>
> colors
> ncolors = !d.table_size - 16
> bottom = 16
>
>
> ; -----
> ; Create a window
> ; -----
>
> if ( keyword_set( gif ) ) then pixmap = 1 else pixmap = 0
>
> if ( not keyword_set( ps ) ) then begin
>   window, /free, xsize = 900, ysize = 600, title = 'Remap Display',
pixmap = pixmap
>   wset, !d.window
> endif
>
>
> ; -----
> ; Read in all the file tags
> ; -----
>
> openr, lun_desc, descriptor, /get_lun
>
> n_files = 0
> this_tag = "
>
> while ( not eof( lun_desc ) ) do begin
>

```

```

>   n_files = n_files + 1
>   readf, lun_desc, this_tag
>   if ( n_files eq 1 ) then $
>     file_tag = [ this_tag ] $
>   else $
>     file_tag = [ file_tag, this_tag ]
>
> endwhile
>
> free_lun, lun_desc
>
>
> ; -----
> ; Loop over files
> ; -----
>
> for i = 0, n_files - 1 do begin
>
>
> ; -----
> ; Read HDR file to determine line/element #s
> ; -----
>
>   hdr_file = file_tag[ i ] + '.HDR'
>   openr, lun_hdr, hdr_file, /get_lun
>
>   cbuf = "
>   line = -1
>   element = -1
>   n_data_files = 1
>
>   while ( not eof( lun_hdr ) ) do begin
>
>     readf, lun_hdr, cbuf
>     c_data_file = '.' + string( n_data_files + 1, format = '(i3.3)' )
>
>     case 1 of
>
>
> ; -----
> ; Get satellite/instrument character ID
> ; -----
>
>     ( strpos( cbuf, 'Satellite' ) ne -1 ): begin
>       sat_inst_id = strtrim( strmid( cbuf, 12, 28 ) )
>     end
>
>
>

```

```

> ; -----
> ; Get image date and start time
> ; -----
>
> ( strpos( cbuf, 'Image Date' ) ne -1 ): begin
>   date = strtrim( strmid( cbuf, 14, 9 ) )
>   time = strtrim( strmid( cbuf, 59, 12 ) )
> end
>
>
> ; -----
> ; Read parameters from first occurrence of data
> ; -----
>
> ( strpos( cbuf, '.001' ) ne -1 ): begin
>   n_data_files = 1
>   parameter = strtrim( strmid( cbuf, 14, 12 ) )
>   band      = fix( strmid( cbuf, 27, 4 ) )
>   n_bands   = 1
>   line      = fix( strmid( cbuf, 32, 7 ) )
>   element   = fix( strmid( cbuf, 40, 8 ) )
>   data_scale = float( strmid( cbuf, 74, 5 ) )
>
>   case 1 of
>     ( strpos( cbuf, 'Short Integer (8 bit)' ) ne -1 ): itype = 1
>     ( strpos( cbuf, 'Integer (16 bit)' ) ne -1 ): itype = 2
>     ( strpos( cbuf, 'Long Integer (32 bit)' ) ne -1 ): itype = 3
>   else: begin
>     message, 'Invalid Element Format', /info
>     close, /all
>     return
>   end
> endcase
>
> end
>
>
> ; -----
> ; Determine how many files are represented
> ; -----
>
> ( strpos( cbuf, c_data_file ) ne -1 ): begin
>   n_data_files = n_data_files + 1
>   parameter = [ parameter, strtrim( strmid( cbuf, 14, 12 ) ) ]
>   band = [ band, fix( strmid( cbuf, 27, 4 ) ) ]
>   if ( band[ n_data_files - 1 ] ne band[ n_data_files - 2 ] ) then
$
>     n_bands = n_bands + 1

```

```

>     data_scale = [ data_scale, float( strmid( cbuf, 74, 5 ) ) ]
>
>     case 1 of
>     ( strpos( cbuf, 'Short Integer (8 bit)' ) ne -1 ): itype =
itype, 1 ]
>     ( strpos( cbuf, 'Integer (16 bit)' ) ne -1 ): itype = [ itype,
2 ]
>     ( strpos( cbuf, 'Long Integer (32 bit)' ) ne -1 ): itype =
itype, 3 ]
>     else: begin
>         message, 'Invalid Element Format', /info
>         close, /all
>         return
>     end
> endcase
> end
>
>
> ; -----
> ; Get LAT and LON scale factors
> ; -----
>
> ( strpos( cbuf, '.LAT' ) ne -1 ): begin
>     lat_scale = float( strmid( cbuf, 74, 5 ) )
> end
>
> ( strpos( cbuf, '.LON' ) ne -1 ): begin
>     lon_scale = float( strmid( cbuf, 74, 5 ) )
> end
>
>
> ; -----
> ; Default
> ; -----
>
>     else:
>
> endcase
>
> endwhile
>
> free_lun, lun_hdr
>
>
> ; -----
> ; Check that dimensional info has been read in
> ; -----
>

```

```

> if ( ( line eq -1 ) or ( element eq -1 ) ) then begin
>   message, 'Invalid LINE and ELEMENT values', /info
>   close, /all
>   return
> endif
>
>
> ; -----
> ; Read in the latitude and longitude information
> ; -----
>
>   ilat = intarr( element, line )
>   openr, lun_lat, file_tag[ i ] + '.LAT', /get_lun
>   readu, lun_lat, ilat
>   free_lun, lun_lat
>   lat = float( ilat ) / lat_scale
>   ilat = 0B
>
>   ilon = intarr( element, line )
>   openr, lun_lon, file_tag[ i ] + '.LON', /get_lun
>   readu, lun_lon, ilon
>   free_lun, lun_lon
>   lon = -1.0 * float( ilon ) / lon_scale
>   ilon = 0B
>
>
> ; -----
> ; Loop over the number of bands
> ; -----
>
>   for j = 0, n_data_files - 1 do begin
>
>
> ; -----
> ; Get current band # and data file name
> ; -----
>
>   this_band = band[ j ]
>   this_data_file = file_tag[ i ] + '.' + string( j + 1, format =
( i3.3 ) )
>   out_file_tag = file_tag[ i ] + '.' + strcompress( sat_inst_id,
/remove_all ) + $
>               '.band' + string( this_band, format = '(i2.2)' ) +
'.' + $
>               parameter[ j ]
>
>
> ; -----

```

```

> ; Only concerned with Albedo, Radiance, and Temperature
> ; -----
>
> if ( ( parameter[ j ] eq 'Albedo' ) or $
>      ( parameter[ j ] eq 'Radiance' ) or $
>      ( parameter[ j ] eq 'Temperature' ) ) then begin
>
>
> ; -----
> ; Read data and scale it
> ; -----
>
> print, format = '( /5x, "Remapping ", a, "..." )', this_data_file
>
> idata = make_array( element, line, type = itype[ j ] )
>
> openr, lun_data, this_data_file, /get_lun
> readu, lun_data, idata
> free_lun, lun_data
> data = float( idata ) / data_scale[ j ]
> idata = 0B
>
>
> ; -----
> ; Call imagemap routine
> ; -----
>
> if ( keyword_set( ps ) ) then begin
>
>     ;- get the current color table
>
>     tvlct, r, g, b, /get
>     ncolors = !d.table_size - 16 & bottom = 16
>     r_old = r( bottom : bottom + ncolors - 1 )
>     g_old = g( bottom : bottom + ncolors - 1 )
>     b_old = b( bottom : bottom + ncolors - 1 )
>
>     ;- change to Postscript mode
>
>     set_plot, 'PS'
>
>     ;- load the interpolated color table, and make sure bottom entry
is black
>
>     ncolors = !d.table_size - 16
>     r_new = congrid( r_old, ncolors, /interp )
>     g_new = congrid( g_old, ncolors, /interp )
>     b_new = congrid( b_old, ncolors, /interp )

```

```

>      tvlct, r_new, g_new, b_new, bottom
>      tvlct, 0, 0, 0, 0
>
>      ;- set Postscript options
>
>      !p.font = 0
>      device, /landscape, bits_per_pixel = 8, /color, /helv, font_size
= 9
>
>      endif
>
>      range = [ min( data ), max( data ) ]
>      erase, 7
>      pos = [ .05, .05, .9, .95 ]
>      map_set, limit = [ 25, -125, 50, -60 ], pos = pos, /noerase
>      title = sat_inst_id + ' Band ' + string( this_band, format =
'(i2)' ) + $
>              ' ' + parameter[ j ] + ' ; ' + date + ' @ ' + time
>      xyouts, 0.5, 0.975, title, /normal, align = 0.5, color = 0,
charsize = 1.5
>      imagemap, data, lat, lon, range = range, /noerase, ncolors =
ncolors, bottom =
>      bottom, $
>              missing = 7B
>      map_continents, /hires, color = 1
>      map_continents, /hires, /usa, color = 1
>      map_grid, label = 1, color = 2
>      plots, [ pos(0), pos(2), pos(2), pos(0), pos(0) ], [ pos(1),
pos(1), pos(3),
>      pos(3), pos(1) ], $
>              /normal, color = 0
>      colorbar, pos = [ .935, .05, .96, .95], title = parameter[ j ],
format = '(f5.1)',
>      $
>              /right, /vertical, min = range[ 0 ], max = range[ 1 ],
divisions = 8, $
>              ncolors = ncolors, bottom = bottom, color = 0
>
>      if ( keyword_set( ps ) ) then begin
>      device, /close
>      spawn, 'mv idl.ps ' + out_file_tag + '.ps'
>      print, format = '( 5x, "File ", a, " created" )', out_file_tag +
'.ps'
>      set_plot, 'X'
>      !p.font = -1
>      tvlct, r_old, g_old, b_old, bottom
>      endif
>

```

```

> if ( keyword_set( gif ) ) then begin
>   tvlct, r, g, b, /get
>   write_gif, out_file_tag + '.gif', tvrd(), r, g, b
>   print, format = '( 5x, "File ", a, " created" )', out_file_tag +
'.gif'
>   endif
>
>   if ( not keyword_set( no_pause ) ) then begin
>     print, format = '( 5x, "Press any key to continue, <x> to
quit..." )'
>     result = get_kbrd( 1 )
>     if ( strlowercase( result ) eq 'x' ) then return
>     endif
>
>   endif else begin
>
>     print, format = '( 5x, "Ignoring ", a, ", Element Format: ", a )',
$
>     this_data_file, parameter[ j ]
>
>   endelse
>
>   endfor
>
> endfor
>
> end
>
> <-----cut here----->
>
>
> --
> Paul van Delst      A little learning is a dangerous thing;
> CIMSS @ NOAA/NCEP   Drink deep, or taste not the Pierian spring;
> Ph: (301)763-8000 x7274 There shallow draughts intoxicate the brain,
> Fax:(301)763-8545    And drinking largely sobers us again.
>                      Alexander Pope.

```
