Subject: Re: Latitude longitude and Image Navigation
Posted by Lavanya on Wed, 22 Jun 2011 11:07:57 GMT
View Forum Message <> Reply to Message

On Jun 22, 9:41 am, AISHWARYA <spacea...@gmail.com> wrote:
> Can someone help me on Navigating the Images in IDL? I have no idea as
> where to start with IDL on Image Navigation. The images were acquired
> from telescope and target was Venus. How can I make a latitude
> longitude grid for venus knowing the North pole position angle and
> north pole position distance?
>
> Any help would be great!
>
> Thank you,
> Aishwarya.

I guess you should register the image, if you are provided with
metadata, then you could create projections in IDL

Subject: Re: Latitude longitude and Image Navigation
Posted by AISHWARYA on Thu, 23 Jun 2011 04:28:24 GMT
View Forum Message <> Reply to Message

On Jun 22, 4:07 pm, Lavanya <lavany...@gmail.com> wrote:
> On Jun 22, 9:41 am, AISHWARYA <spacea...@gmail.com> wrote:
>
>> Can someone help me on Navigating the Images in IDL? I have no idea as
>> where to start with IDL on Image Navigation. The images were acquired
>> from telescope and target was Venus. How can I make a latitude
>> longitude grid for venus knowing the North pole position angle and
>> north pole position distance?
>
>> Any help would be great!
>
>> Thank you,
>> Aishwarya.
>
> I guess you should register the image, if you are provided with
> metadata, then you could create projections in IDL

Thank you Lavanya. But, I do not have the Metadata. What I have is the
raw pixel coordinate, Sub observer Lat long and centre pixel
coordinate of venus disc. I would require something like a
transformation matrix and Im looking for the parameters of the same. I
might have to first fix the North pole through the above mentioned
North pole position angle and North pole distance and then interpolate
to find the Latlong for toher pixels.

Regards,
Aishwarya.

Subject: Re: Latitude longitude and Image Navigation
Posted by Brian Wolven on Thu, 23 Jun 2011 17:22:18 GMT
View Forum Message <> Reply to Message

Back when I was doing work of this type with the Galilean satellites I used the NAIF SPICE library to generate the position and orientation of each body relative to the observer based on the timetags in the images. That may be overkill for you. On the other hand, somebody may already have a nice package of routines out there ready to use with IDL for this purpose. Just a thought.

http://naif.jpl.nasa.gov/naif/toolkit_IDL.html

Subject: Re: Latitude longitude and Image Navigation
Posted by AISHWARYA on Fri, 24 Jun 2011 04:25:05 GMT
View Forum Message <> Reply to Message

On Jun 23, 10:22 pm, Brian Wolven <brian.wol...@gmail.com> wrote:
> Back when I was doing work of this type with the Galilean satellites I used the NAIF SPICE library to generate the position and orientation of each body relative to the observer based on the timetags in the images. That may be overkill for you. On the other hand, somebody may already have a nice package of routines out there ready to use with IDL for this purpose. Just a thought.
>
> http://naif.jpl.nasa.gov/naif/toolkit_IDL.html

Yeah I tried NAIF-SPICE library and also mailed them. But I was told that they dont work more on FITS data and Telescopic images. I'm trying my hands on SOHO image navigation code which I think might throw some light on Image Navigation processes in IDL.

But, May I know if you were using the North pole angle and distance in your method? If so, can you expatiate on how did you use them for Image Navigation?

Thank you,
Aishwarya.

Subject: Re: Latitude longitude and Image Navigation
Posted by Brian Wolven on Fri, 24 Jun 2011 17:08:37 GMT
View Forum Message <> Reply to Message

As I recall, I was generating a lat-long grid to overlay on HST/STIS images of Galilean satellites. It looks like I used the North Pole position vector and the sub-observer point (calculated in SPICE) to get the viewing orientation and then drew the projected lat-long grid based on that information. This was a long time ago (15 years?), albeit in galaxy not so far away. I think the code below is what I used to calculate the rotation matrix for the grid; I don't know if it will help or not. Looks like I first rotated the z-axis to be aligned with the pole vector, then found the rotation about the pole to place the sub-observer point at the proper coordinates.

```
function polar_rotation,body
 ;===========================================================
====================
; Rotate z-axis into specified pole position vector (nppv)
 ;===========================================================
====================
  nppv  = fourvector(unit_vector(body.nppv)) ; unit 4-vector along NPPV
  zaxis = [0.,0.,1.,0.]
  rpol1 = rotate_a2b(zaxis,nppv)
 ;===========================================================
====================
; Latitude must be converted from planetographic to planetocentric coordinates.
; Planetocentric co-latitude is equivalent to theta in spherical coordinates.
; -1 * longitude is equivalent to phi in spherical coordinates.
 ;===========================================================
====================
  axisratio = body.r_polar/body.r_equatorial
  sublatg   = body.subobsvrlat
  sublatc   = pg2pc_lat(sublatg,axisratio)
  subcolat  = (90.-sublatc)*!dtor                ; theta in spherical coords
  sublon    = -1.*body.subobsvrlon*!dtor         ; phi   in spherical coords
 ;===========================================================
====================
; Rotate body such that line of sight intersects specified subobserver point.
; Rotate only in the plane perpendicular to the pole (nppv)
 ;===========================================================
====================
  subdes =  [cos(sublon)*sin(subcolat),sin(sublon)*sin(subcolat),cos(sub colat)]
  subcur = -1.*body.pos/length(body.pos) ;
  oldang = angle_between(subdes,zaxis)
  bodang = angle_between(subcur,nppv)
  subdes = fourvector(subdes)#rpol1      ;4-vector of desired sub-observer pt
  subcur = fourvector(subcur)           ;4-vector of initial sub-observer pt
  newang = angle_between(subdes,nppv)
  newdes = subdes - nppv*dot_product(nppv,subdes) ; Project components on plane
  newsub = subcur - nppv*dot_product(nppv,subcur) ;   perpendicular to nppv.
  rpol2  = rotate_a2b(newdes,newsub)
  finang = angle_between(subdes#rpol2,nppv)
 ;===========================================================
====================
; Diagnostics
 ;============================================================
====================
; print,'Curr: ',float(reform(subcur))
; print,'Dsrd: ',float(reform(subdes))
;  print,float(reform([sublon/!dtor,sublatg/!dtor,sublatc,subco lat/!dtor]))
; print,float([bodang,oldang,newang,finang])
 ;============================================================
====================
; Return total matrix to calling procedure.
 ;============================================================
====================
  return,rpol1#rpol2
end
```

Subject: Re: Latitude longitude and Image Navigation
Posted by AISHWARYA on Fri, 24 Jun 2011 18:51:08 GMT

View Forum Message <> Reply to Message

On Jun 24, 10:08 pm, Brian Wolven <brian.wol...@gmail.com> wrote:
> As I recall, I was generating a lat-long grid to overlay on HST/STIS images of Galilean satellites. It looks like I used the North Pole position vector and the sub-observer point (calculated in SPICE) to get the viewing orientation and then drew the projected lat-long grid based on that information. This was a long time ago (15 years?), albeit in galaxy not so far away. I think the code below is what I used to calculate the rotation matrix for the grid; I don't know if it will help or not. Looks like I first rotated the z-axis to be aligned with the pole vector, then found the rotation about the pole to place the sub-observer point at the proper coordinates.

```
>
>  function polar_rotation,body
>   ;============================================================
===============-=====
>  ; Rotate z-axis into specified pole position vector (nppv)
>   ;============================================================
===============-=====
>    nppv  = fourvector(unit_vector(body.nppv)) ; unit 4-vector along NPPV
>    zaxis = [0.,0.,1.,0.]
>    rpol1 = rotate_a2b(zaxis,nppv)
>   ;============================================================
===============-=====
>  ; Latitude must be converted from planetographic to planetocentric coordinates.
>  ; Planetocentric co-latitude is equivalent to theta in spherical coordinates.
>  ; -1 * longitude is equivalent to phi in spherical coordinates.
>   ;============================================================
===============-=====
>    axisratio = body.r_polar/body.r_equatorial
>    sublatg   = body.subobsvrlat
>    sublatc   = pg2pc_lat(sublatg,axisratio)
>    subcolat  = (90.-sublatc)*!dtor                 ; theta in spherical coords
>    sublon    = -1.*body.subobsvrlon*!dtor          ; phi   in spherical coords
>   ;=============================================================
===============-=====
>  ; Rotate body such that line of sight intersects specified subobserver point.
>  ; Rotate only in the plane perpendicular to the pole (nppv)
>   ;=============================================================
===============-=====
>    subdes =  [cos(sublon)*sin(subcolat),sin(sublon)*sin(subcolat),cos(sub colat)]
>    subcur = -1.*body.pos/length(body.pos) ;
>    oldang = angle_between(subdes,zaxis)
>    bodang = angle_between(subcur,nppv)
>    subdes = fourvector(subdes)#rpol1      ;4-vector of desired sub-observer pt
>    subcur = fourvector(subcur)            ;4-vector of initial sub-observer pt
>    newang = angle_between(subdes,nppv)
>    newdes = subdes - nppv*dot_product(nppv,subdes) ; Project components on plane
>    newsub = subcur - nppv*dot_product(nppv,subcur) ;   perpendicular to nppv.
>    rpol2  = rotate_a2b(newdes,newsub)
>    finang = angle_between(subdes#rpol2,nppv)
>   ;=============================================================
===============-=====
>  ; Diagnostics
>   ;=============================================================
===============-=====
>  ; print,'Curr: ',float(reform(subcur))
>  ; print,'Dsrd: ',float(reform(subdes))
>  ;  print,float(reform([sublon/!dtor,sublatg/!dtor,sublatc,subco lat/!dtor]))
>  ; print,float([bodang,oldang,newang,finang])
>   ;=============================================================
===============-=====
>  ; Return total matrix to calling procedure.
>   ;=============================================================
===============-=====
>    return,rpol1#rpol2
>  end
```

Thank you, i'm trying to understand the code. But,is there any way
that I can measure the pixel correpsonding to North pole angle and
distance by any inbuilt function? Does plate scale come into picture
anywhere?

Method 2 : I was trying my hands on Map_Image. I could fix the centre
but lat min and Lat max seems to be a question. Do you think Map_image
can help me in this?

Also, I recently found a SOHO code written for converting camera
coordinates to solar coordinates by Dr.Iwang

this is the link : http://www.astro.washington.edu/docs/idl/cgi-bin/getpro/libr
ary32.html?FIND_LIMB2

I was trying a similar one but IDL stops running when the first while
loop is encountered.

Regards,
Aishwarya.

Subject: Re: Latitude longitude and Image Navigation
Posted by AISHWARYA on Fri, 24 Jun 2011 19:14:20 GMT

View Forum Message <> Reply to Message

On Jun 24, 10:08 pm, Brian Wolven <brian.wol...@gmail.com> wrote:

> As I recall, I was generating a lat-long grid to overlay on HST/STIS images of Galilean satellites. It looks like I used the North Pole position vector and the sub-observer point (calculated in SPICE) to get the viewing orientation and then drew the projected lat-long grid based on that information. This was a long time ago (15 years?), albeit in galaxy not so far away. I think the code below is what I used to calculate the rotation matrix for the grid; I don't know if it will help or not. Looks like I first rotated the z-axis to be aligned with the pole vector, then found the rotation about the pole to place the sub-observer point at the proper coordinates.

```
>
>  function polar_rotation,body
>   ;===========================================================
===============-=====
>  ; Rotate z-axis into specified pole position vector (nppv)
>   ;===========================================================
===============-=====
>    nppv  = fourvector(unit_vector(body.nppv)) ; unit 4-vector along NPPV
>    zaxis = [0.,0.,1.,0.]
>    rpol1 = rotate_a2b(zaxis,nppv)
>   ;===========================================================
===============-=====
>  ; Latitude must be converted from planetographic to planetocentric coordinates.
>  ; Planetocentric co-latitude is equivalent to theta in spherical coordinates.
>  ; -1 * longitude is equivalent to phi in spherical coordinates.
>   ;===========================================================
===============-=====
>    axisratio = body.r_polar/body.r_equatorial
>    sublatg   = body.subobsvrlat
>    sublatc   = pg2pc_lat(sublatg,axisratio)
>    subcolat  = (90.-sublatc)*!dtor                ; theta in spherical coords
>    sublon    = -1.*body.subobsvrlon*!dtor         ; phi   in spherical coords
>   ;===========================================================
===============-=====
>  ; Rotate body such that line of sight intersects specified subobserver point.
>  ; Rotate only in the plane perpendicular to the pole (nppv)
>   ;===========================================================
===============-=====
>    subdes =  [cos(sublon)*sin(subcolat),sin(sublon)*sin(subcolat),cos(sub colat)]
>    subcur = -1.*body.pos/length(body.pos) ;
>    oldang = angle_between(subdes,zaxis)
>    bodang = angle_between(subcur,nppv)
>    subdes = fourvector(subdes)#rpol1      ;4-vector of desired sub-observer pt
>    subcur = fourvector(subcur)            ;4-vector of initial sub-observer pt
>    newang = angle_between(subdes,nppv)
>    newdes = subdes - nppv*dot_product(nppv,subdes) ; Project components on plane
>    newsub = subcur - nppv*dot_product(nppv,subcur) ;   perpendicular to nppv.
>    rpol2  = rotate_a2b(newdes,newsub)
>    finang = angle_between(subdes#rpol2,nppv)
>   ;===========================================================
===============-=====
>  ; Diagnostics
>   ;===========================================================
===============-=====
>  ; print,'Curr: ',float(reform(subcur))
>  ; print,'Dsrd: ',float(reform(subdes))
>  ;  print,float(reform([sublon/!dtor,sublatg/!dtor,sublatc,subco lat/!dtor]))
>  ; print,float([bodang,oldang,newang,finang])
>   ;===========================================================
===============-=====
>  ; Return total matrix to calling procedure.
>   ;===========================================================
===============-=====
>    return,rpol1#rpol2
>  end
```

Guess, this code will give me some ideas. Can you tell me what do
'subdes'subcur', oldang', 'bodyang'refer to ?

Thank you,
Aishwarya.

Subject: Re: Latitude longitude and Image Navigation
Posted by penteado on Mon, 27 Jun 2011 23:10:17 GMT
View Forum Message <> Reply to Message

On Jun 24, 1:25 am, AISHWARYA <spacea...@gmail.com> wrote:

> Yeah I tried NAIF-SPICE library and also mailed them. But I was told
> that they dont work more on FITS data and Telescopic images. I'm
> trying my hands on SOHO image navigation code which I think might
> throw some light on Image Navigation processes in IDL.

The kind of file you use is irrelevant. The SPICE toolkit has the routines and data files to calculate geometry between many solar system objects.