Image Geo-Registration

Overview

The Image Registration algorithm calculates geo-location information for each pixel in an OCAMS image.

History

Initial version - 3 Dec 2013

Algorithm Description

This algorithm uses the SPICE libraries to calculate the location in latitude and longitude of each pixel in an OCAMS image as well as the illumination angles (phase, incident and emission) of each pixel. This information will be based on the most recently delivered shape model for Bennu, the reconstructed spacecraft trajectory (SP kernel) and the pointing information (quaterions) from the spacecraft downlinked to the ground with the observations.

This algorithm may be implemented using IDL, c or Fortran using standard SPICE libraries with SPICE kernels appropriate for Bennu and it's current mission phase. An example IDL program that uses the SPICE libraries to calculate this type of information is appended in the "Example" section. We will need to modify this example to take unregistered OCAMS images as input and return geo-registered OCAMS images as ISIS cubes. ISIS cube are the preferred file format for the ISIS image mosaic routines as well as the photometric modeling and correction function being developed by the Photometric modeling group.

Parameters

Input Parameters

- An unregistered level 2 OCAMS image (in FITS format)
- CK kernel created from the quaterions downlinked from the spacecraft
- A reconstructed SPK kernel from NAV
- The most recent shape model of Bennu

Output Parameters

- A geo-registered OCAMS image (in ISIS cub format)

Example

```
PRO fovint, inst=inst, inputtime=inputtime, lat=lat,lon=lon,phase=phase, incid=incid,
emissn=emissn
  META_KERNEL = './kernels/remote/remote_sensing_fovint.tm'
  SCID = 'OSIRIS_REX'
  TARGET = 'BENNU'
  FRAME = 'J2000'
  INST = inst
  MAXBND = 4
  CORRECT = 'LT+S'
  TYPE = 'PLANETOCENTRIC'
  vecnam  = ['Boundary Corner 1', $
            'Boundary Corner 2', $
            'Boundary Corner 3', $
            'Boundary Corner 4', $
            'OSIRIS-REx OCAMS Boresight' ]
  cspice_furnsh, META_KERNEL
  cspice_ktotal, 'all', n_kernels
  if n_kernels lt 9 then begin
    printf, 'Error loading kernels'
    list_all_kernels
    stop
  endif
  if not keyword_set(inputtime) then begin
    inputtime = "   ; Define as a string
    READ, inputtime, PROMPT='Enter Time: '
  endif


  print, "Converting UTC Time: ", inputtime
  cspice_str2et, inputtime, et
  print, FORMAT = '(A, F16.6)', "   ET Seconds Past J2000: ", et
  cspice_bodn2c, INST, code, found
  if ( found ) then begin
    cspice_getfov, code, MAXBND, shape, frame, bsight, bounds
  endif else begin
     print, 'No code corresponding to name: ', name
  endelse
  for  i=0, MAXBND do begin


     if ( i lt MAXBND ) then begin
          dvec = bounds(*,i)
     endif
```

```
    if ( i eq MAXBND )  then begin
         dvec = bsight
    endif


    cspice_sincpt, 'Ellipsoid', TARGET, et, 'IAU_BENNU', CORRECT, SCID, $
              frame, dvec, spoint, trgepc, srfvec, found
    if ( found ) then begin
      print, 'Vector: ', vecnam[i]
      print, FORMAT='(A,3(F16.3))',  'Position vector of surface intercept in the IAU_BENNU
frame (km):', spoint


      dist = norm( srfvec );


      cspice_reclat, spoint,  radius, lon, lat
      lon = lon * cspice_dpr()
      lat = lat * cspice_dpr()
      cspice_ilumin, 'Ellipsoid', TARGET, et, 'IAU_BENNU', CORRECT, SCID, spoint, trgepc,
srfvec, phase, solar, emissn
      phase  = phase *cspice_dpr()
      solar  = solar *cspice_dpr()
      emissn = emissn*cspice_dpr()


      print, 'Planetocentric coordinates of the intercept:'
      print, ' '
      print, FORMAT = '("     Planetocentric Latitude  (deg) = ",F18.10)', lat
      print, FORMAT = '("     Planetocentric Longitude (deg) = ",F18.10)', lon
      print, FORMAT = '("     Phase angle              (deg) = ",F18.10)', phase
      print, FORMAT = '("     Solar angle              (deg) = ",F18.10)', solar
      print, FORMAT = '("     Emission angle           (deg) = ",F18.10)', emissn
      if ( i eq MAXBND )  then begin
         lon  = lon * cspice_rpd()
         cspice_bodn2c, TARGET, code, found
         if (found) then begin
           cspice_et2lst, et, code, lon, TYPE, hr,  mn, sc , time, ampm
           print,"
           print, FORMAT = '("Local Solar Time at boresight intercept ",A)', ampm
         endif
      endif
      print, ' '
```

```
    endif else begin
      print, 'Intercept not found.'
      return
    endelse


  endfor
  cspice_kclear
  cspice_ktotal, 'all', n_kernels
end
```