XXVIII CANARY ISLANDS WINTER SCHOOL OF ASTROPHYSICS

SOLAR SYSTEM EXPLORATION

An Introduction to ISIS3 Software

Julia de León
Instituto de Astrofísica de Canarias - IAC

November 11, 2016. Friday 15:00 – 17:30

CREATE A DIRECTORY TO STORE YOUR DATA:

> mkdir /scratch/WSFriday11
> cd /scratch/WSFriday11
> cp –r /home/jmlc/WS2016/ISISExercise .
> cd ISISExercise
> gedit Openme.txt &

ISIS - Integrated Software for Imagers and Spectrometers

Developed by the US Geological Survey (USGS) for NASA

- Over 300 image processing applications
- Strong emphasis on geometric functionality
  - Photogrammetry / Camera models
  - Cartography / Map projections
  - Adjust/Improve Instrument Position and Orientation
  - Generate Digital Map Mosaics
- In use for over 30 years
  - Software generations: PICS, ISIS2, ISIS3
- Support for over 55 NASA/ESA instruments
  - Framing Cameras (e.g, MDIS Narrow Angle and Wide Angle cameras)
  - Line Scan Cameras (e.g., THEMIS-IR, HiRISE)
  - Push Frame Cameras (e.g., LROC-WAC)
  - Special Cases
    - Radar Instruments (e.g., LROC-MiniRF)
    - Spot Instruments (e.g., Cassini-VIMS)

Mission Instruments Supported by ISIS3

Moon
- Lunar Orbiter III, IV, &, V (Medium and HiRes)
- Clementine UVVIS, NIR, HIRES, & LWIR
- Apollo Metric 15/16/17
- Apollo Panoramic 15/16/17
- Lunar Reconnaissance Orbiter NACL, NACR, WAC (VIS & UV), MiniRF
- Chandrayaan-1 M3, MiniRF
- Kaguya MI (VIS & NIR)

Mercury
- Mariner 10 (A & B)
- MESSENGER MDIS (NAC & WAC)

Asteroids
- Dawn FC (1 & 2), VIR
- Near Earth Asteroid Rendezvous Shoemaker MSI
- Hayabusa AMICA
- Upcoming – OSIRIS-REx OCAMS

Multiple Target Bodies
- Voyager 1 & 2 (NAC & WAC)

Mars
- Mars Global Surveyor MOC (NAC & WAC)
- Mars Odyssey THEMIS (VIS & IR)
- Mars Express HRSC
- Mars Reconnaissance Orbiter HiRISE, CTX, MARCI, CRISM
- Viking Orbiter 1 & 2 (A & B)

Jovian
- Galileo SSI

Saturnian
- Cassini ISS (NAC & WAC), VIMS

Other
- Ideal Camera (Special ISIS Perfect Virtual Instrument – Distortion-Free!)

Pluto
- New Horizons MVIC, LORRI, LEISA

Mission Instruments Supported by ISIS3

Moon
- Lunar Orbiter III, IV, &, V (Medium and HiRes)
- Clementine UVVIS, NIR, HIRES, & LWIR
- Apollo Metric 15/16/17
- Apollo Panoramic 15/16/17
- Lunar Reconnaissance Orbiter NACL, NACR, WAC (VIS & UV), MiniRF
- Chandrayaan-1 M3, MiniRF
- Kaguya MI (VIS & NIR)

Mercury
- Mariner 10 (A & B)
- MESSENGER MDIS (NAC & WAC)

Asteroids
- Dawn FC (1 & 2), VIR
- Near Earth Asteroid Rendezvous Shoemaker MSI
- Hayabusa AMICA
- Upcoming – OSIRIS-REx OCAMS

Multiple Target Bodies
- Voyager 1 & 2 (NAC & WAC)

Mars
- Mars Global Surveyor MOC (NAC & WAC)
- Mars Odyssey THEMIS (VIS & IR)
- Mars Express HRSC
- Mars Reconnaissance Orbiter HiRISE, CTX, MARCI, CRISM
- Viking Orbiter 1 & 2 (A & B)

Jovian
- Galileo SSI

Saturnian
- Cassini ISS (NAC & WAC), VIMS

Other
- Ideal Camera (Special ISIS Perfect Virtual Instrument – Distortion-Free!)

Pluto
- New Horizons MVIC, LORRI, LEISA

ISIS3 Documentation, Support and User Guides

- **General Information**

  http://isis.astrogeology.usgs.gov

- **Installation Guide**

  http://isis.astrogeology.usgs.gov/documents/InstallGuide

- **Table of ISIS Applications**

  http://isis.astrogeology.usgs.gov/Application

  The ISIS3 Software Manual is organized by Functional Category & Mission Specific Programs

- **User Support Forum**

  http://isis.astrogeology.usgs.gov/IsisSupport

- **Online Workshops**

  http://isis.astrogeology.usgs.gov/IsisWorkshop

- **Isis Command Line Options**

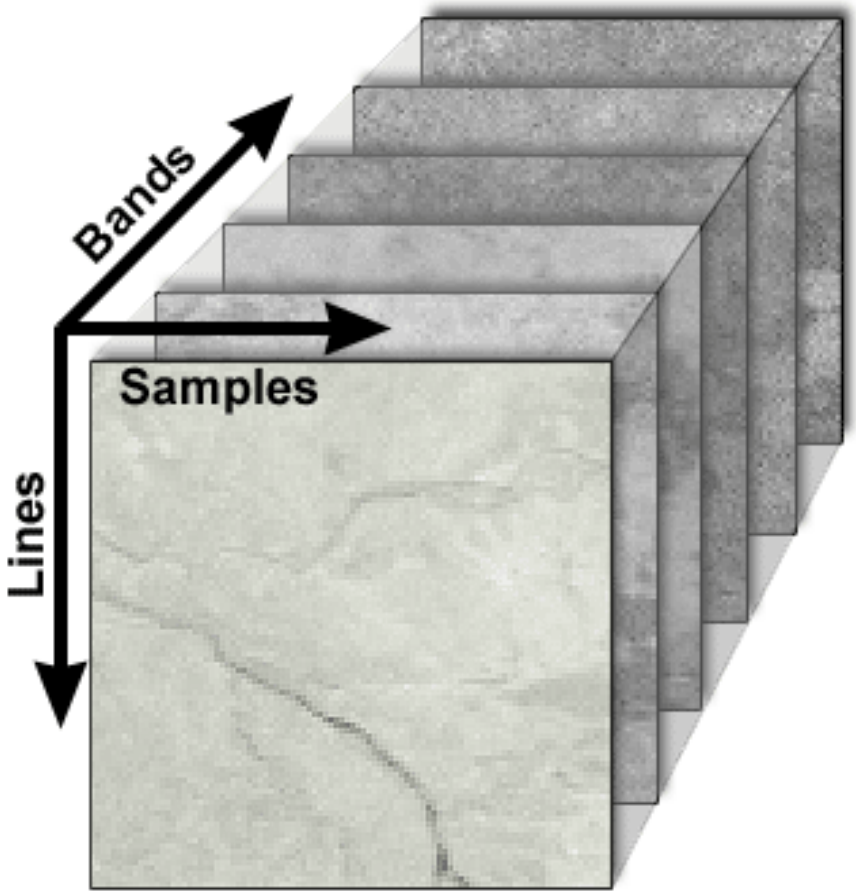
  http://isis.astrogeology.usgs.gov/documents/CommandLine/CommandLine.html

- **Unix/Linux Command Reference Cheat-Sheet**

  http://www.cheat-sheets.org/saved-copy/fwunixref.pdf

ISIS3 data format

ISIS3 uses **CUBES**. A cube is a 3D image with axis: samples, lines, and bands. Sample and line dimensions are used to represent spatial information while band dimension represents spectral information.



Know your data

You have downloaded 3 images from the PDS:

FC21B0007101_11273005958F3G.IMG F3=749 nm
FC21B0007102_11273010009F4G.IMG F4=917 nm
FC21B0007106_11273010035F8G.IMG F8=438 nm

The first step is to transform the images into ISIS cubes:

ascii2isis	ISIS preserves the labels of the original data set (PDS, VICAR, FITS)
fits2isis	
pds2isis	
raw2isis	
vicar2isis	
std2isis (png, jpeg, tiff)	

```
> ls -1 *.IMG | sed 's/\.IMG//' > base.lis
> more base.lis
    FC21B0007101_11273005958F3G
    FC21B0007102_11273010009F4G
    FC21B0007106_11273010035F8G
> dawnfc2isis -batchlist=base.lis from=\$1.IMG to=\$1.cub
> ls
    FC21B0007101_11273005958F3G.IMG   FC21B0007106_11273010035F8G.cub
    FC21B0007101_11273005958F3G.cub   Openme.txt
    FC21B0007102_11273010009F4G.IMG   base.lis
    FC21B0007102_11273010009F4G.cub   print.prt
    FC21B0007106_11273010035F8G.IMG   vesta_gaskell_512_110825_dem.cub
```

Know your data

You can know get information about your images. We can read the header of the cube using ***catlab***:

```
> catlab from=FC21B0007101_11273005958F3G.cub to=Imagelabel.txt
> more Imagelabel.txt
    Object = IsisCube
        Object = Core
            StartByte   = 65537
            Format      = Tile
            TileSamples = 512
            TileLines   = 512

            Group = Dimensions
              Samples = 1024
              Lines   = 1024
              Bands   = 1
            End_Group
    ....
    ....
```

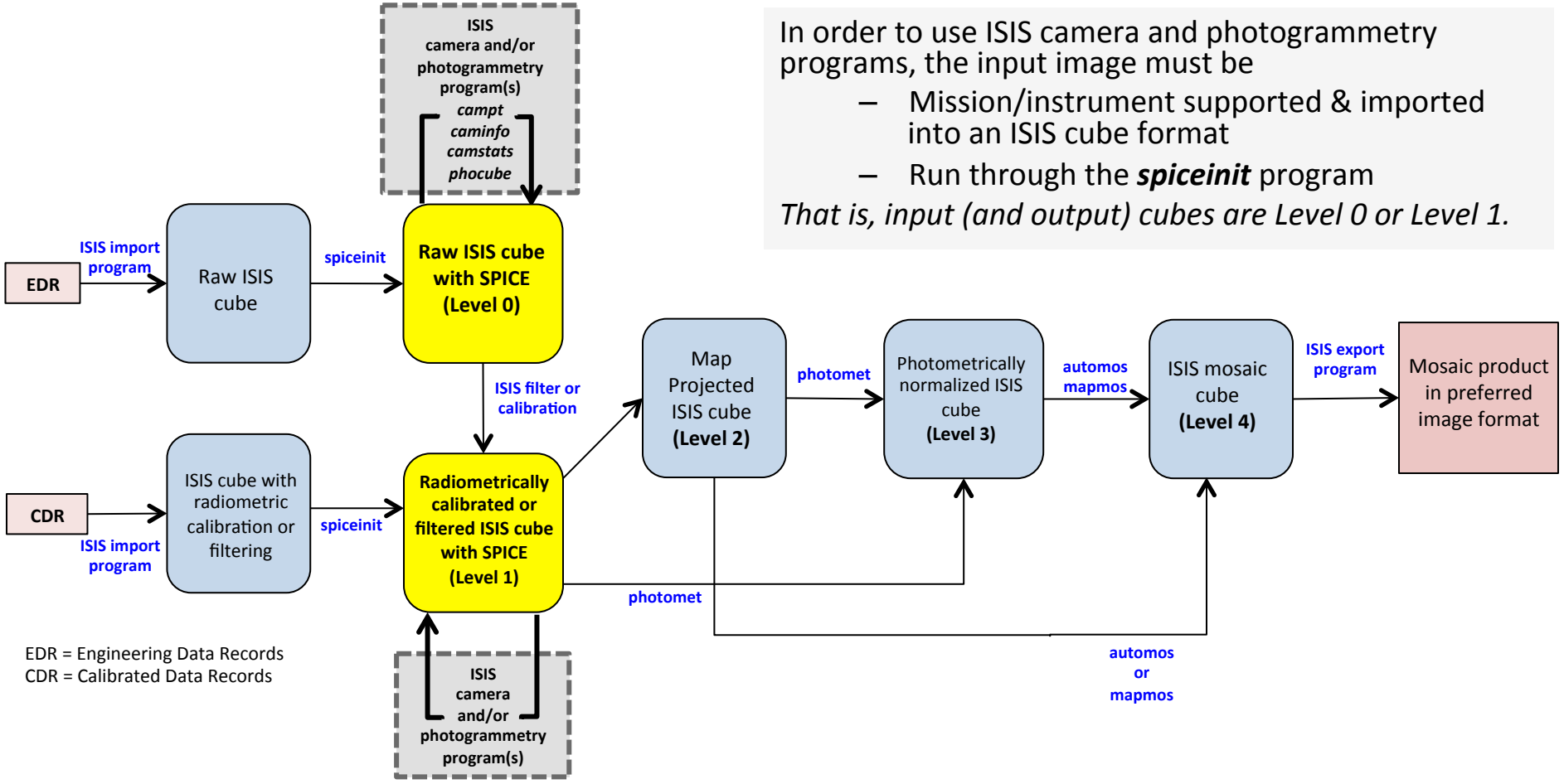
ISIS3 camera and photogrammetry

Cartographic Processing – ISIS Cubes with SPICE

In order to use ISIS camera and photogrammetry programs, the input image must be

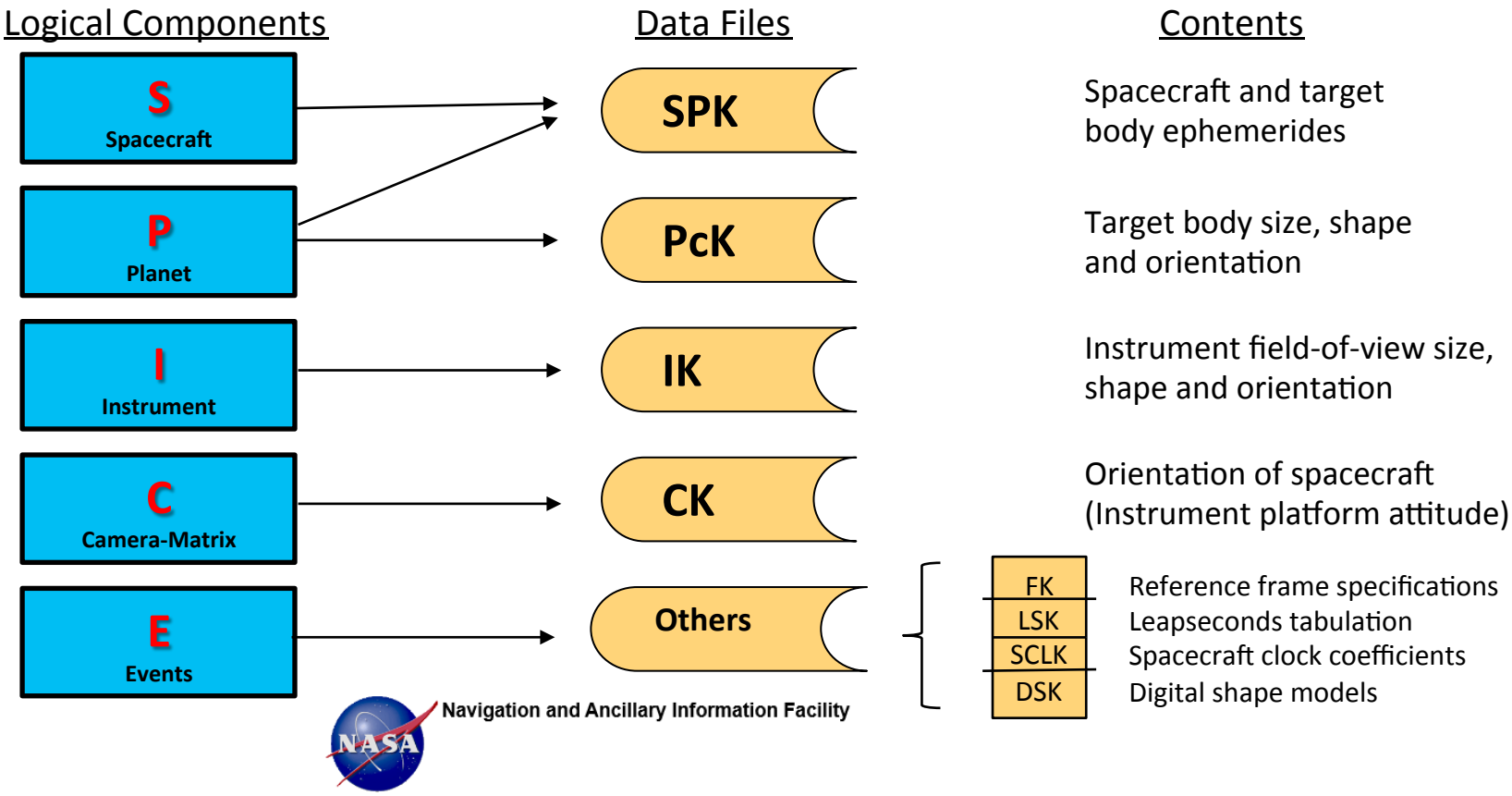
- Mission/instrument supported & imported into an ISIS cube format
- Run through the ***spiceinit*** program

That is, input (and output) cubes are Level 0 or Level 1.



ISIS3 camera and photogrammetry

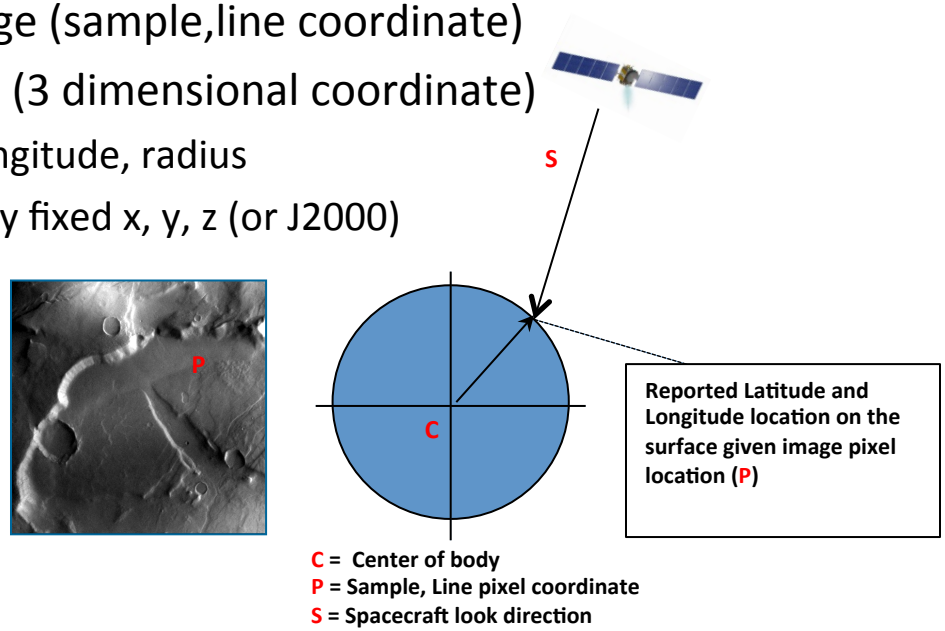
SPICE kernels contain ephemeris and orientation data needed for cartographic processing…



ISIS3 camera and photogrammetry

Camera Model

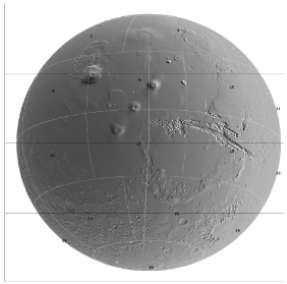
- Simple definition of a camera model
  - Given a pixel location on an image (sample,line coordinate)
  - Compute a location on a surface (3 dimensional coordinate)
    - Usually reported as latitude, longitude, radius
    - Can also be represented as body fixed x, y, z (or J2000)

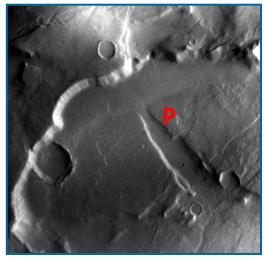


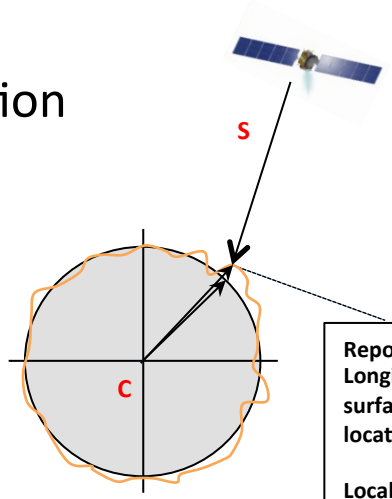
ISIS3 camera and photogrammetry

Camera Model

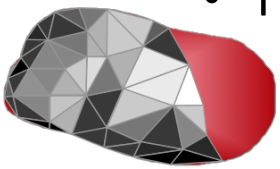
- Digital Shape Model
  - Allows for accurate calculation and reporting of local geometric and photometric values
  - Allows for orthorectification in a map projection
    - Digital Elevation Model
      - Such as the Moon, Mars, Vesta
        - » LOLA for Moon
        - » MOLA for Mars







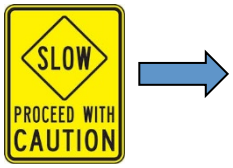
    - Triangulated Irregular Network
      - For small, irregular bodies
      - Such as asteroids and small satellites

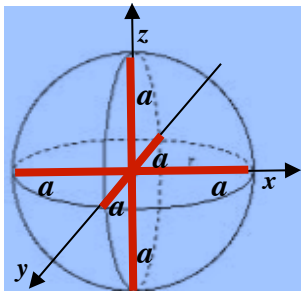


ISIS3 camera and photogrammetry

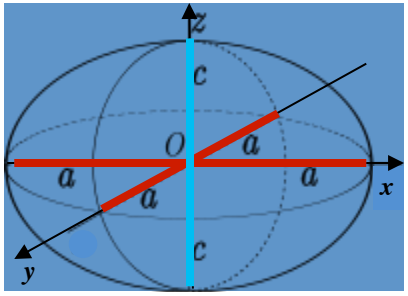
Camera Model

- Target body shape
  - Radius
    - A = Major Equatorial Axis
    - B = Minor Equatorial Axis
    - C = Minor Polar Axis

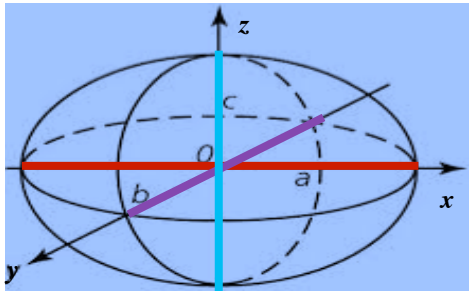




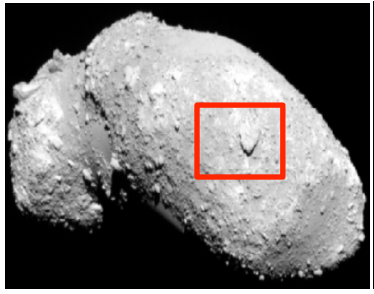
Sphere
A = **B** = **C**
Ex: Moon



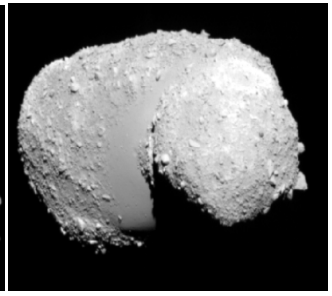
Spheroid
A = **B** > **C**
Ex: Mars



Triaxial Ellipsoid
A > **B** > **C**
Ex: Io, Vesta



Local Region on an Irregular Body
A > **B** > **C**
Ex: 25143 Itokawa



Global Irregular Body
Ex: 25143 Itokawa

*ISIS3 cartographic functionality is accurate in computing surface points until mapping an entire global irregular body shape

Prepare your data

Add the information. Assign the DEM shape model and associated PcK. ***spiceinit*** adds camera pointing and spacecraft info to an image cube*:*

```
> spiceinit -batchlist=base.lis from=\$1.cub shape=user
  model=vesta_gaskell_512_110825_dem.cub

> catlab from=FC21B0007101_11273005958F3G.cub
```

Write camera information from a single point location with ***campt***. Default location is the center of the image (center sample, center line):

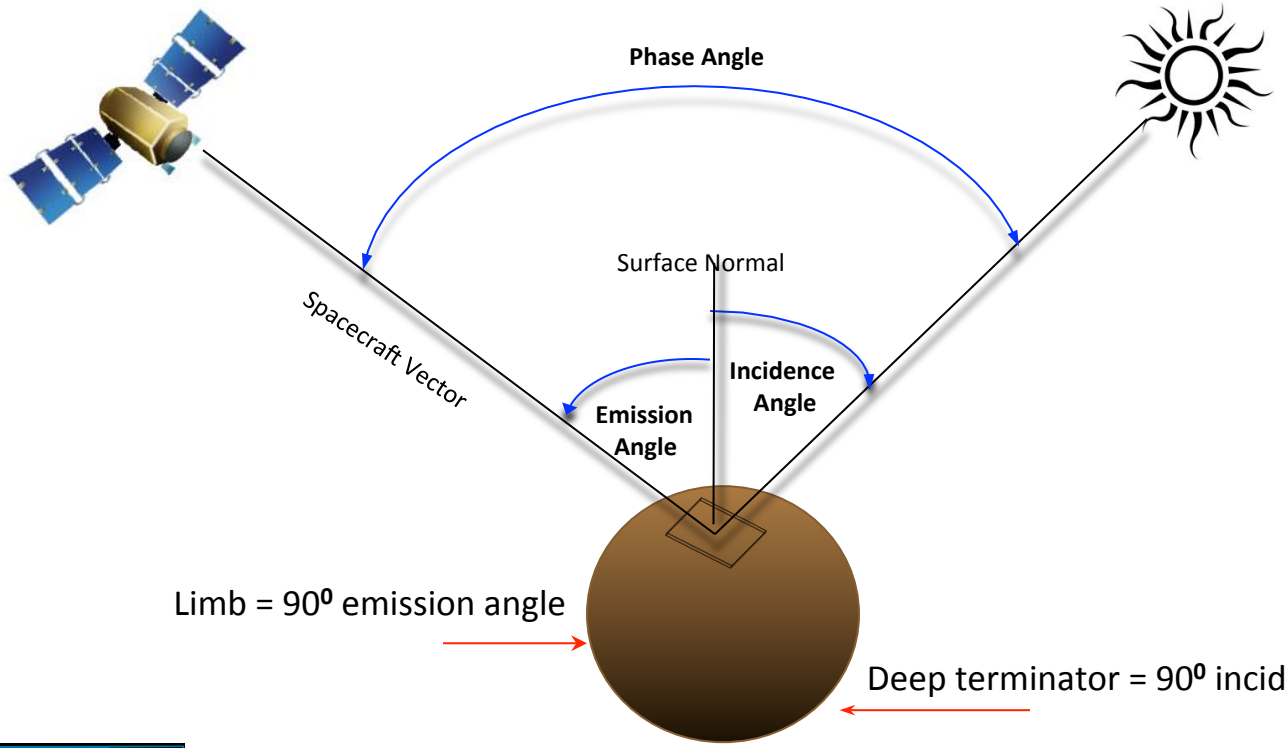
```
> campt from=FC21B0007101_11273005958F3G.cub
    Group = GroundPoint
        Filename                = FC21B0007101_11273005958F3G.cub
        Sample                  = 512.0
        Line                    = 512.0
        PixelValue              = 0.014663695
        RightAscension          = 299.91282664987
        Declination             = -41.463904971556
        PlanetocentricLatitude  = -6.4539240564576
....
....
```

Output of *campt*

```
Group = GroundPoint
  Filename                  = FC21B00(…).cub
  Sample                    = 512.0
  Line                      = 512.0
  PixelValue                = 0.014663695
  RightAscension            = 299.91282664987
  Declination               = -41.463904971556
  PlanetocentricLatitude    = -6.4539240564576
  PlanetographicLatitude    = -10.213079601788
  PositiveEast360Longitude  = 95.628007855414
  PositiveEast180Longitude  = 95.628007855414
  PositiveWest360Longitude  = 264.37199214459
  PositiveWest180Longitude  = -95.628007855414
....
....
....
```

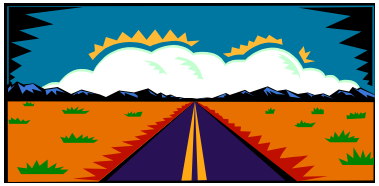
```
# Sun Information
 SunPosition              = (-195038429.19133,
                             235611233.96377,
                             -154736770.45296 )<km>
 SubSolarAzimuth          = 0.24941892418741
 SolarDistance            = 2.2913216599523 <AU>
 SubSolarLatitude         = -26.834892087208
 SubSolarLongitude        = 129.61787820634
 SubSolarGroundAzimuth    = 126.2224441414

 # Illumination and Other
 Phase                    = 37.593420444518
 Incidence                = 38.195989152649
 Emission                 = 0.63279342927108
 NorthAzimuth             = 234.04657257415

 # Time
 EphemerisTime            = 370616465.36427 <seconds>
 UTC                      = 2011-09-30T00:59:59.1819234
 LocalSolarTime           = 9.7340086432713 <hour>
 SolarLongitude           = 282.93728883161
```

Photometric Angles (reported in degrees)



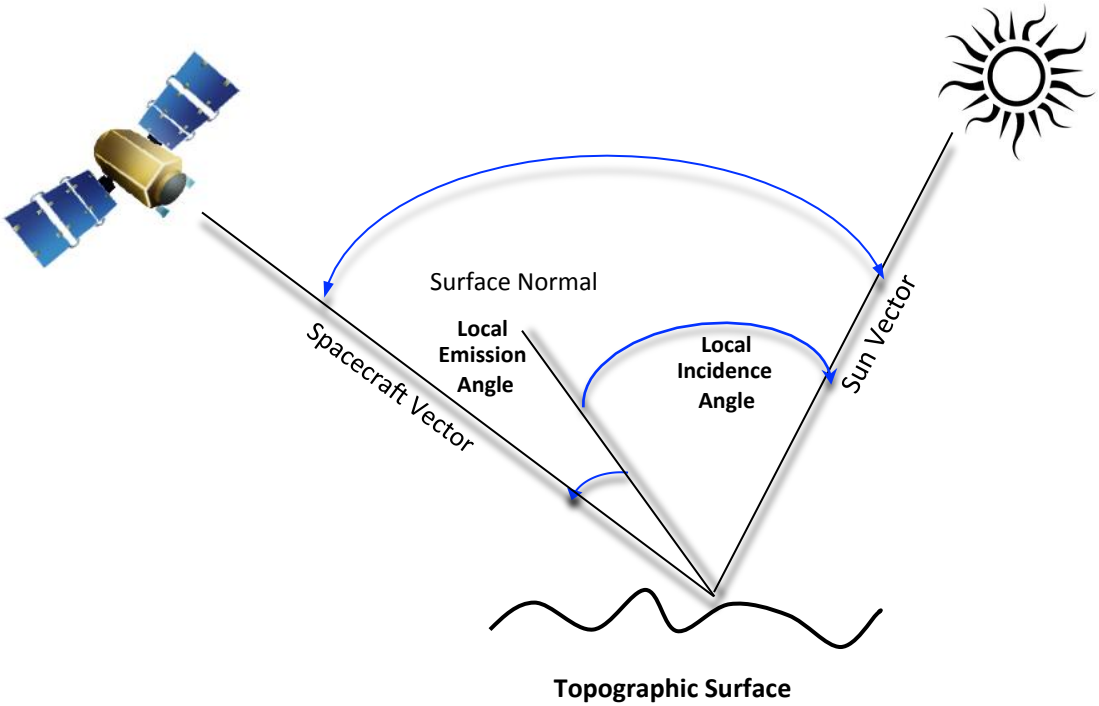
Limb = 90^0 emission angle

Deep terminator = 90^0 incidence angle



Horizon= 90^0 emission angle

Photometric Angles (reported in degrees)

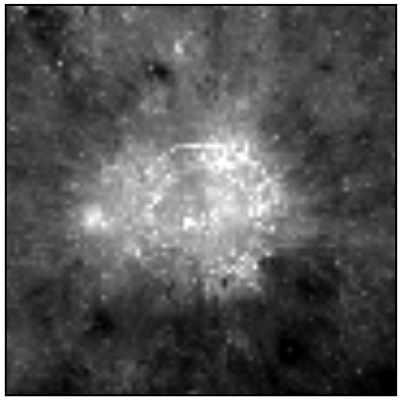


With a DEM shape model, a “local” emission and incidence angle is computed at the pixel point of interest

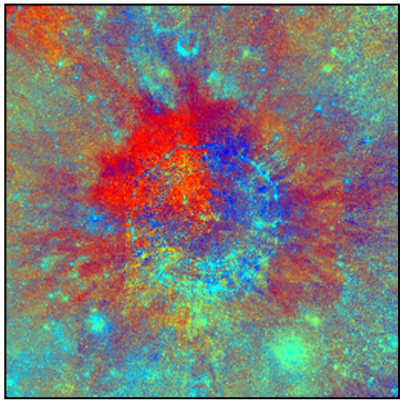
SOLAR SYSTEM EXPLORATION

Clementine Image (750 nm)
Copernicus

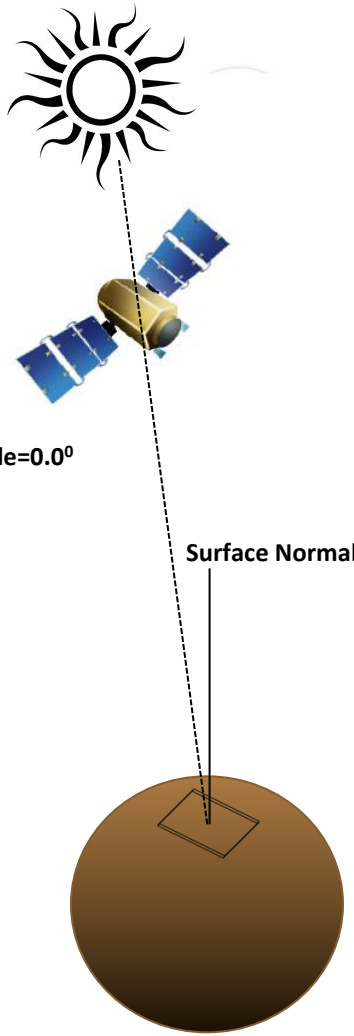
Phase Angle = ~0^0



- No shadows
- No visible topography

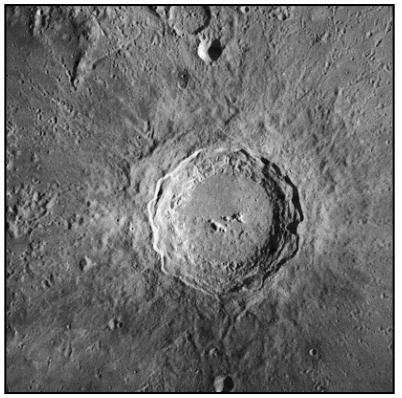


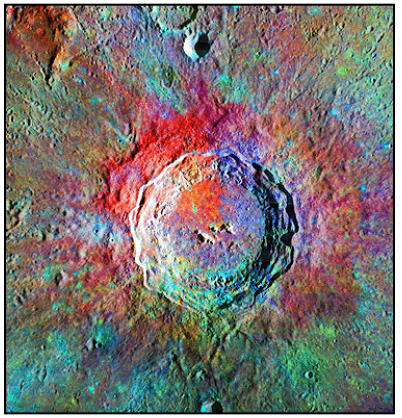
Clementine Multi-Spectral
Color Ratio of Copernicus



Lunar Orbiter IV
Copernicus

Phase Angle = ~65^0

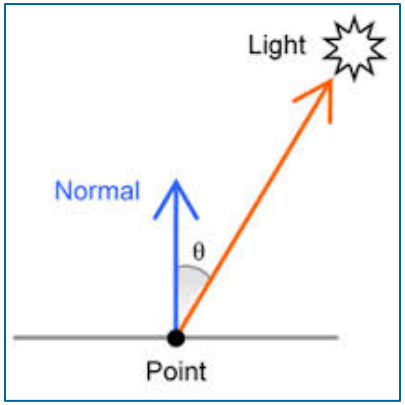




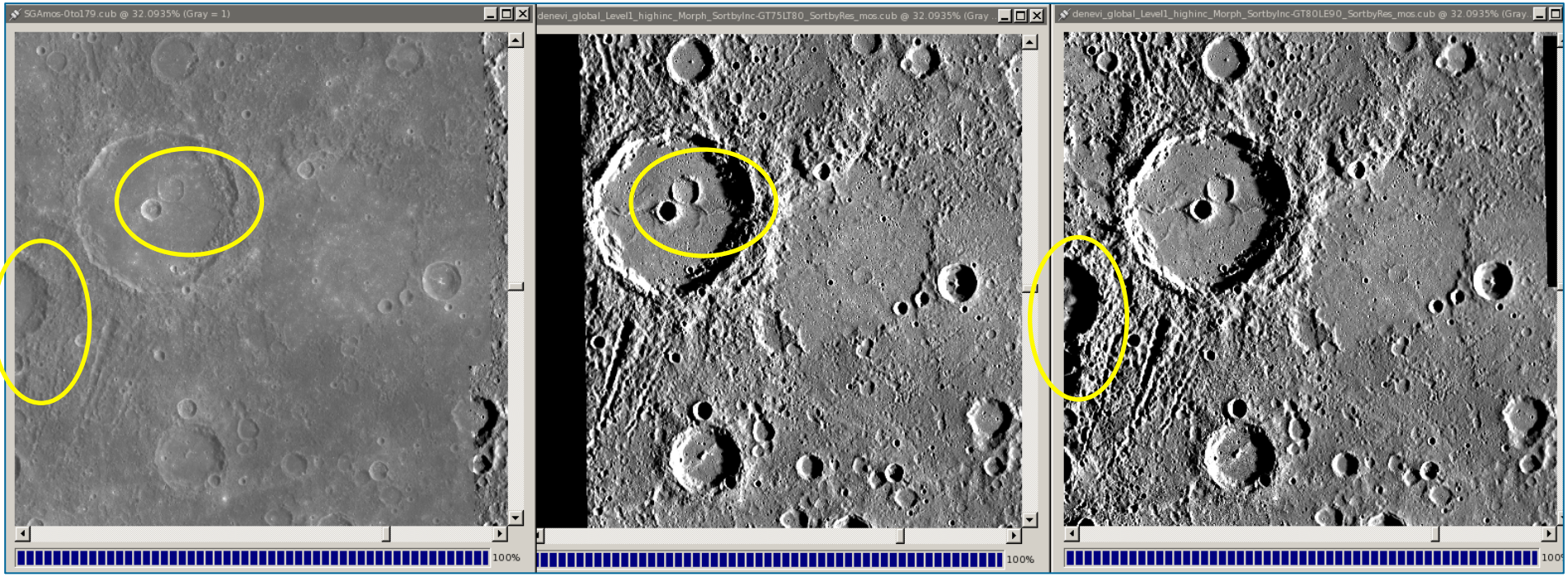
Merge of Lunar Orbiter IV
Clementine Multi-Spectral

Shadows are great, but…

- Incidence angles above 80^0 result in data loss to deep shadows
- Low incidence angles emphasize albedo over morphology



Portions of regional mosaics containing Messenger MDIS images of Mercury



Incidence Angle ~30-50^0 | Incidence Angle 75-80^0 | Incidence Angle 80 -90^0

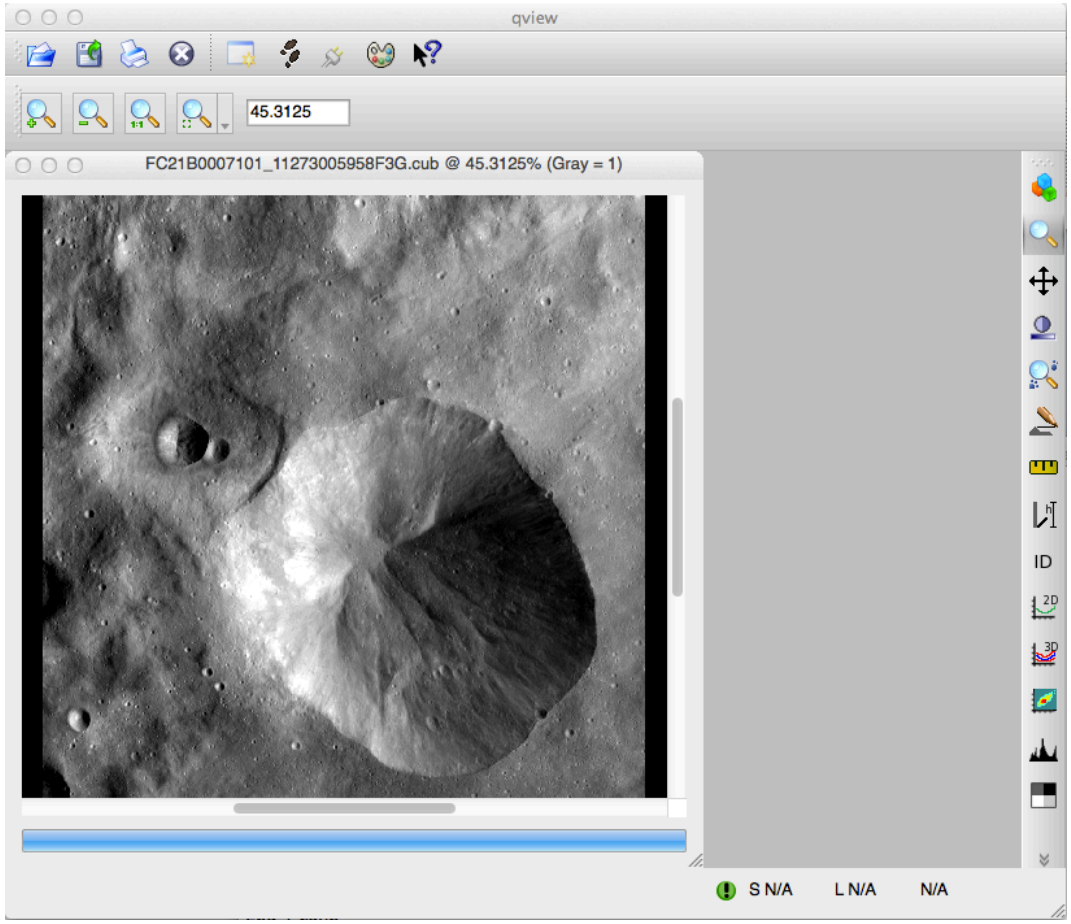
Prepare your data

Add information. Create a text file with geometry statistics and attach the info to image labels:

```
> camstats -batchlist=base.lis from=\$1.cub attach=true linc=10 sinc=10
  to=Camstats.txt

    Group = "User Parameters"
        Filename = FC21B0007106_11273010035F8G.cub
        Linc     = 10
        Sinc     = 10
    End_Group

    Group = Latitude
        LatitudeMinimum           = -16.022001309368
        LatitudeMaximum           = 3.1458418800811
        LatitudeAverage           = -6.7427587623976
        LatitudeStandardDeviation = 3.9389142289714
    End_Group

....
....
```

View your data!

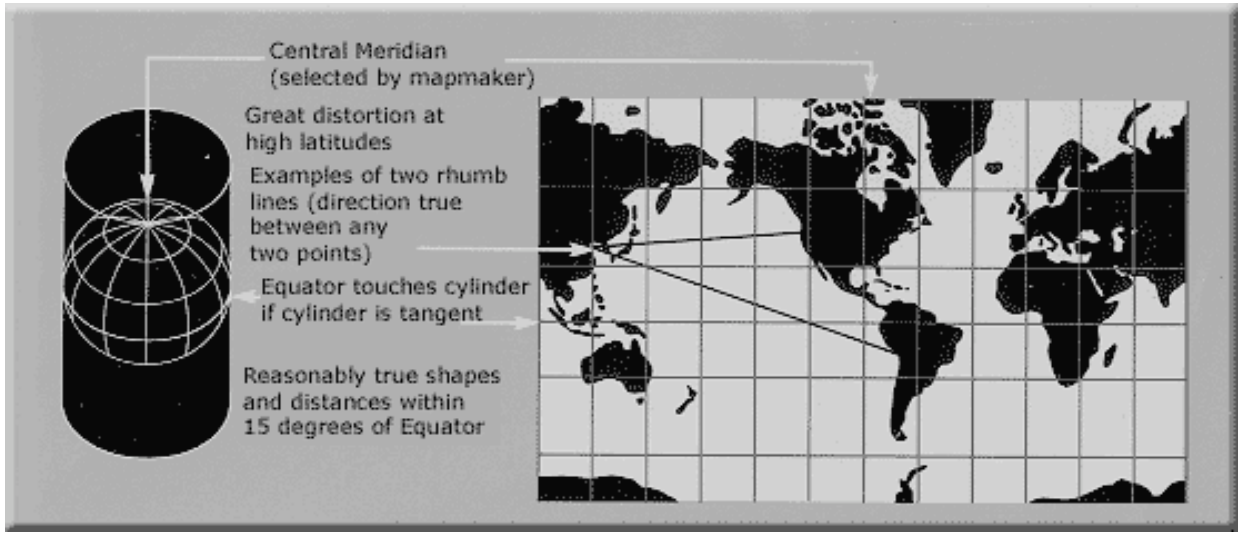
```
> qview FC21B0007101_11273005958F3G.cub &
```



ISIS3 map projecting

Map Products

- Definition of map
  - A representation of a three dimensional target such as a sphere, ellipsoid or an irregular shaped body onto a plane (two dimensional object such as paper or an image)
  - A map projection is an algorithm or equations for transforming a (latitude, longitude) of a three dimensional object (planet/asteroid) into a two dimensional coordinate (x, y)

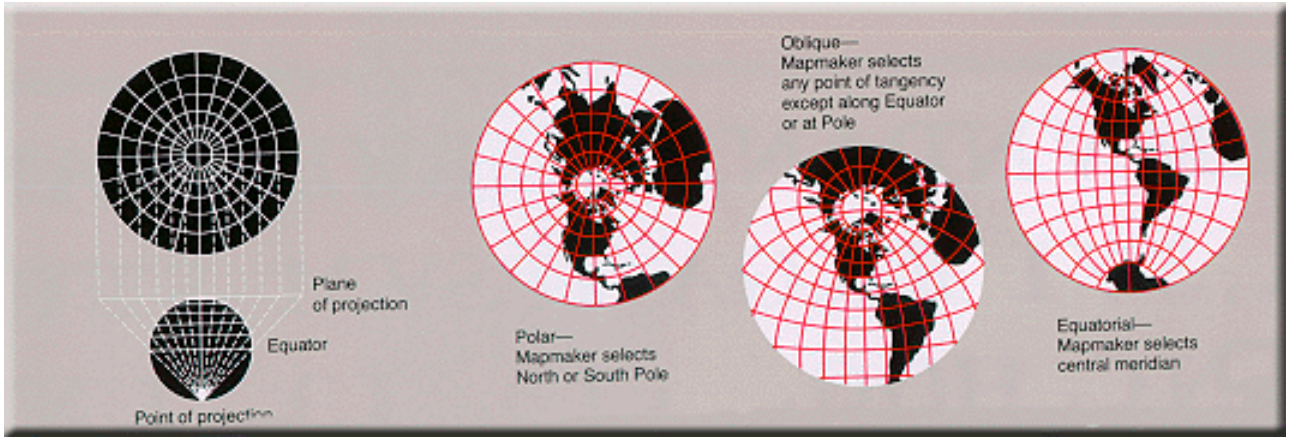


Mercator: x = longitude – centerMerdian y = ln (tan(latitude) + sec(latitude))

ISIS3 map projecting

Supported Map Projections in ISIS3

- Equirectangular
- Lambert Azimuthal Equal Area
- Lambert Conformal
- Lunar Azimuthal Equal Area
- Mercator
- Oblique Cylindrical
- Orthographic
- Point Perspective
- Polar Stereographic
- Robinson
- Simple Cylindrical
- Sinusoidal
- Transverse Mercator



A stereographic projection is conformal, but not equal area or equidistant.

Map projecting your images

Use ***mosrange*** to create a map that will be used to map projecting the images:

```
> ls *G.cub > range.lis
> mosrange fromlist=range.lis to=equi.map projection=equirectangular
> more equi.map
    Group = Mapping
  ProjectionName           = equirectangular
  TargetName               = VESTA
  EquatorialRadius         = 289000.0 <meters>
  PolarRadius              = 229000.0 <meters>
  LatitudeType             = Planetocentric
  LongitudeDirection       = PositiveEast
  LongitudeDomain          = 360
  PixelResolution          = 63.490794422177 <meters/pixel>
  Scale                    = 79.444612154699 <pixels/degree>
  MinPixelResolution       = 62.579930663823 <meters>
  MaxPixelResolution       = 64.401658180531 <meters>
  CenterLongitude          = 95.37037359494
  CenterLatitude           = -6.26669581363
  MinimumLatitude          = -16.032326107132
....
....
```

Map projecting your images

Use ***cam2map*** to map projecting images. We use the already created equi.map for the first image, and then this projected image as the map for the following:

```
> cam2map from=FC21B0007101_11273005958F3G.cub
  to=FC21B0007101_11273005958F3G-CO-EQ.cub map=equi.map pixres=map
  lonseam=continue

  cam2map from=FC21B0007102_11273010009F4G.cub
  to=FC21B0007102_11273010009F4G-EQ.cub map=FC21B0007101_11273005958F3G-CO-EQ.cub
  matchmap=true

  cam2map from=FC21B0007106_11273010035F8G.cub
  to=FC21B0007106_11273010035F8G-EQ.cub map=FC21B0007101_11273005958F3G-CO-EQ.cub
  matchmap=true
```

Compare the non-projected and the projected images:

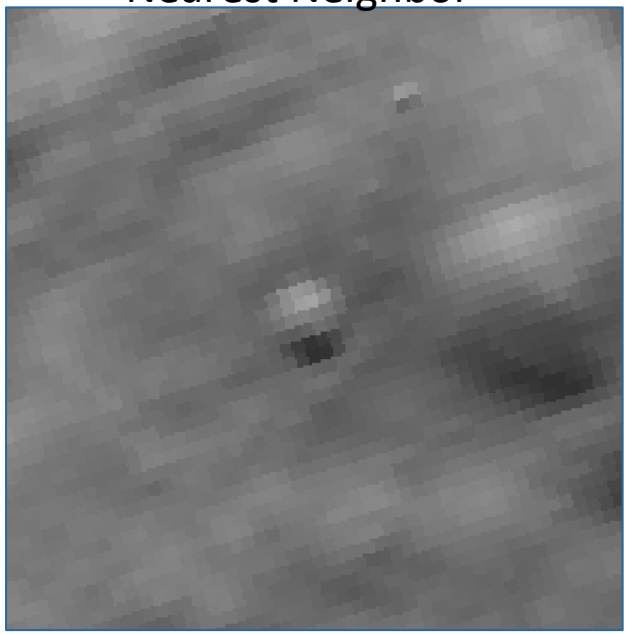
```
> qview FC21B0007102_11273010009F4G.cub FC21B0007102_11273010009F4G-EQ.cub
```

cam2map Interpolation

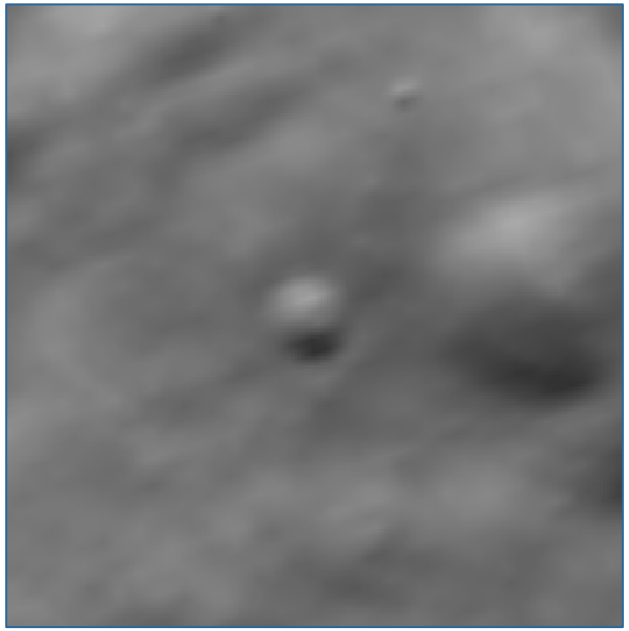
What pixel value to place into the output cube when the input position is not in the center of a pixel?

- **Nearest Neighbor** – Use the DN nearest the position
- **Bi-Linear** – Use the 4 nearest input pixels to linear interpolate a new DN
- **Cubic** – Use the 16 nearest input pixels and a polynomial fit to interpolate a new DN

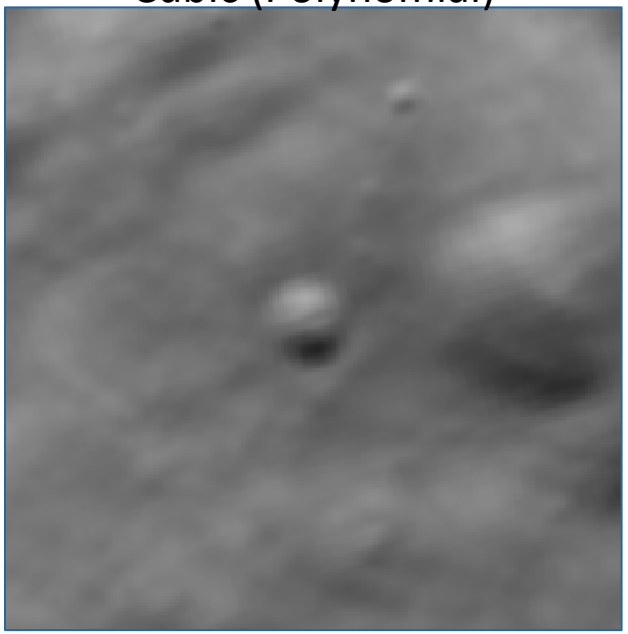
Nearest Neighbor



Bi-Linear



Cubic (Polynomial)



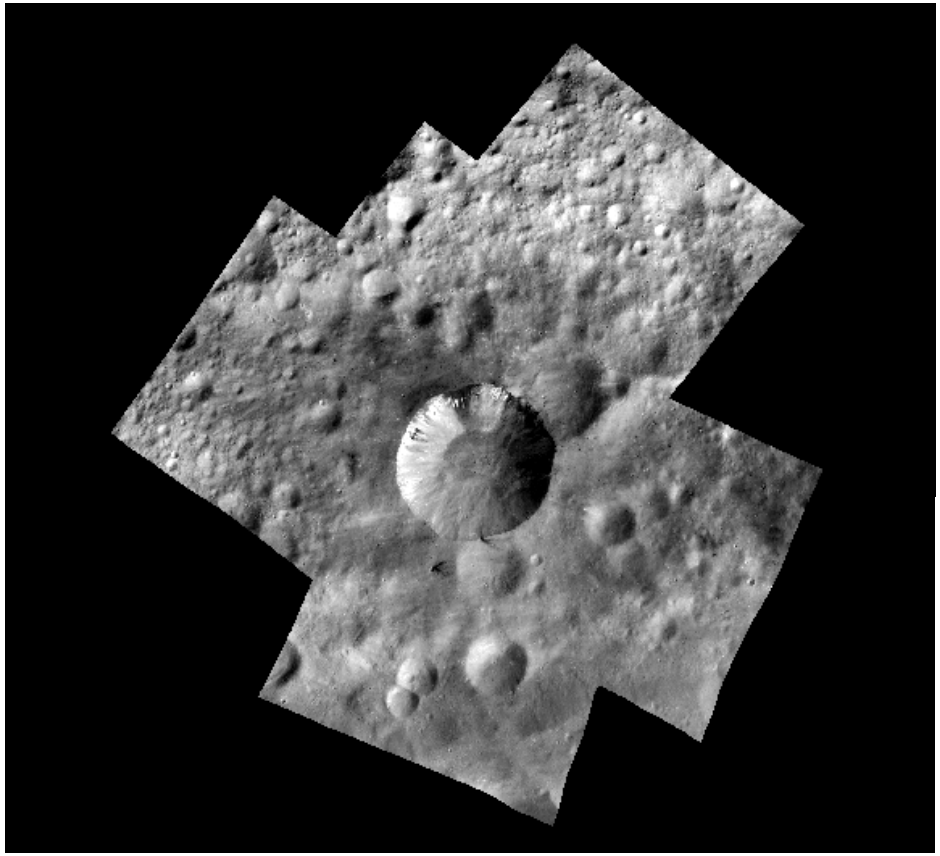
Co-registering your images

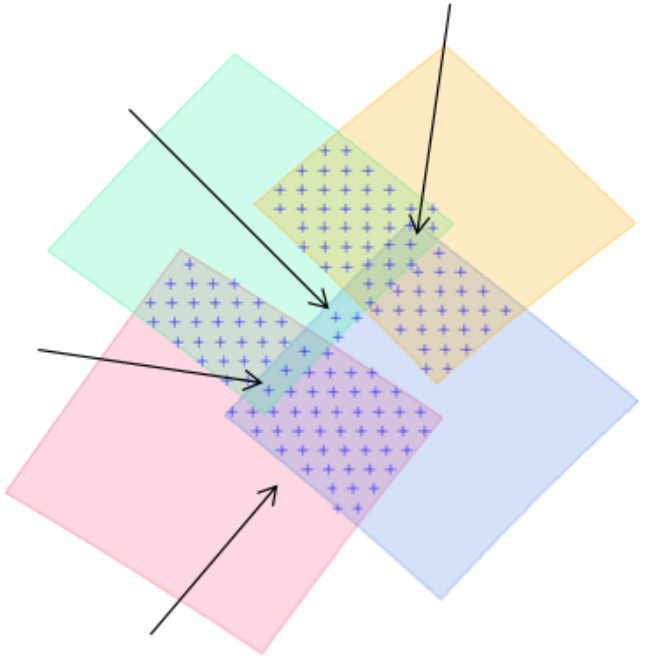
Use ***coreg*** to align all the images. The program computes local translations spaced evenly throughout the image:

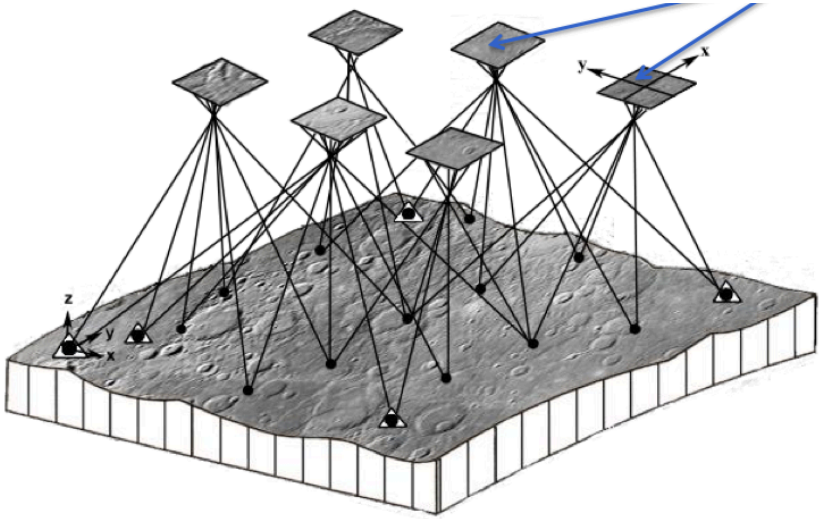
```
> coreg from=FC21B0007102_11273010009F4G-EQ.cub
  match=FC21B0007101_11273005958F3G-CO-EQ.cub to=FC21B0007102_11273010009F4G-CO.cub
  deffile=coreg.def onet=FC21B0007102_11273010009F4G.co.net transform=translate
  interp=bilinear rows=32 columns=32
```

```
Object = AutoRegistration
  Group = Algorithm
    Name = MaximumCorrelation
    Tolerance = 0.85
    SubpixelAccuracy = True
  End_Group

  Group = PatternChip
    Samples = 10
    Lines = 10
  End_Group
....
....
```

Numisia crater on Vesta







ISIS3 photometry

- **Definition**
  - Photometric normalization tools in ISIS are often used for the common purpose of adjusting the brightness and contrast of images obtained under different illumination and viewing geometries so they appear as if obtained under uniform conditions
    - The observed result is a "flattening" of the image brightness that varies as a function of emission and incidence angles
  - To determine or derive the best photometric function and parameters, it is required to perform some level of analysis
    - Best photometric model (target based)
    - Phase function
    - Atmospheric model, if any

ISIS3 photometry

- Photometric Angles
  - **Emission Angle** = The angle between the spacecraft and a vector drawn perpendicular to the planet's surface (surface normal)
  - **Incidence Angle** = The angle between the sun and the surface normal
  - **Phase Angle** = The angle between the sun, the point on the planet surface, and the spacecraft
- ISIS programs compute these angles from the ellipsoid and the elevation shape model (e.g., local slopes) at every pixel
- A photometric model is applied (Lommell-Seeliger, Minnaert, Lambert, Lunar Lambert, Hapke-Henyey) using ***photomet***.
- ***photomet*** performs photometric correction on image pixels acquired under different illumination and viewing geometries by adjusting the brightness and contrast such that the resulting image appears as if obtained under uniform conditions.

Apply photometric correction

Use ***photomet*** to perform photometric correction for each filter. We need to input appropriate Published values (these are from Jian-Yang Li et al. 2013)

```
> photomet -batchlist=base.lis from=$(sed -n '3p' base.lis)-CO.cub
  to=$(sed -n '3p' base.lis)-pho.cub phtname=hapkehen theta=18.1 wh=0.408
  hg1=0.233 hg2=1 hh=0.07 b0=1.7 zerob0standard=false normname=albedo
  incref=30.0 incmat=0.0 thresh=10e30 albedo=1.0 USEDEM=TRUE

    Group = NormalizationModelParametersUsed
         NORMNAME = ALBEDO
         INCREF   = 30.0
         THRESH   = 1.0e+31
         ALBEDO   = 1.0
    End_Group

    Group = PhotometricModelParametersUsed
         PHTNAME        = HAPKEHEN
         THETA          = 18.1
         WH             = 0.408
         HH             = 0.07
         B0             = 1.7
         ZEROB0STANDARD = FALSE
         HG1            = 0.233
         HG2            = 1.0
    End_Group
```

Apply photometric correction

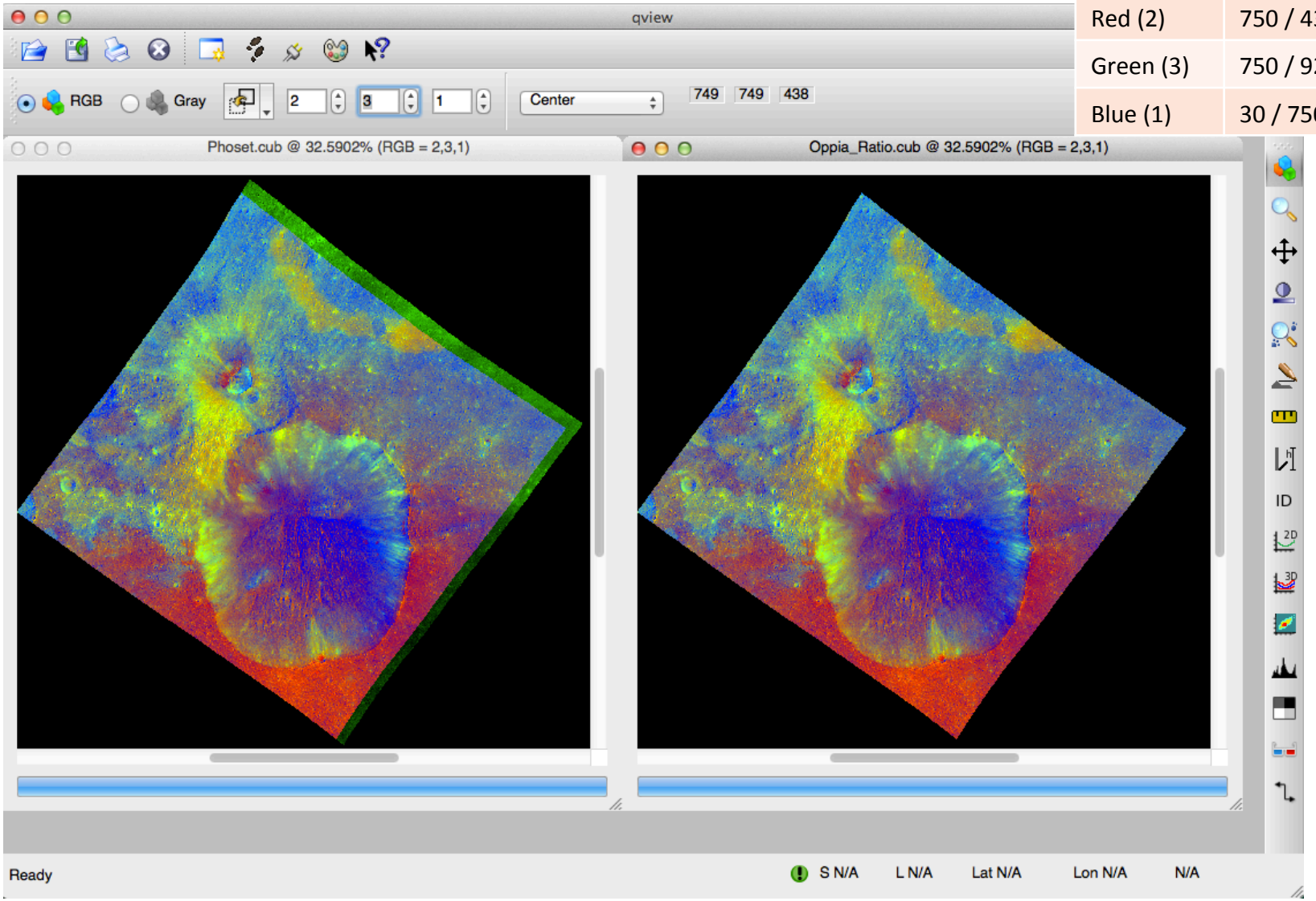
We now divide the images to create different color ratios using ***algebra***:

```
> ls *pho.cub > pho.lis

> algebra from=$(sed -n '1p' pho.lis) from2=$(sed -n '3p' pho.lis)
  to=750over430.cub operator=divide
```

And we put all the ratios into one single cube, trimming the areas that do not overlap:

```
> ls *over*.cub > Ratset.lis

> cubeit from=Ratset.lis to=Phoset.cub

> bandtrim from=Phoset.cub to=Oppia_Ratio.cub
```

Channel	Ratio
Red (2)	750 / 430
Green (3)	750 / 920
Blue (1)	30 / 750



#THIS EXAMPLE SHOWS THE PROCESS FOR USING ISIS TO CREATE A COLOR-RATIO IMAGE FROM DAWN IMAGES OF VESTA
#CODE WRITTEN BY CHRISTOPHER LUKE HAWLEY AND MODIFIED BY JULIA DE LEON

TO MAKE ISIS3 WORK, YOU HAVE TO WRITE THE FOLLOWING LINES:

```
echo $SHELL
```

FOR BASH SHELL:

```
export ISISROOT="/net/japon/scratch/ISIS3/isis"
source ${ISISROOT}/scripts/isis3Startup.sh
```

FOR TCSH SHELL:

```
setenv ISISROOT "/net/japon/scratch/ISIS3/isis"
source ${ISISROOT}/scripts/isis3Startup.csh
```

==========

LETS START!!!!!

#THIS CREATES A BASE NAME LIST OF FILES, WITH NO EXTENSIONS

```
ls -1 *.IMG | sed 's/\.IMG//' > base.lis
```

NOW GET THE APPROPRIATE FORMAT TO BE USED WITH ISIS3:

dawnfc2isis -batchlist=base.lis from=\$1.IMG to=\$1.cub

KNOW YOUR DATA: READ THE HEAD OF ONE IMAGE:

catlab from=FC21B0007101_11273005958F3G.cub to=ImageLabel.txt

ASSIGN THE DEM SHAPEMODEL AND ASSOCIATED PCK (Planetary Constants Kernel). spiceinit ADDS CAMERA POINTING AND SPACECRAFT INFO TO AN IMAGE CUBE:

spiceinit -batchlist=base.lis from=\$1.cub shape=user model=vesta_gaskell_512_110825_dem.cub

KNOW YOUR DATA. WRITE CAMERA INFORMATION FROM A SINGLE POINT LOCATION

campt from=FC21B0007101_11273005958F3G.cub

KNOW YOUR DATA. CREATE A TEXT FILE WITH GEOMETRY STATISTICS

camstats -batchlist=base.lis from=\$1.cub attach=true linc=10 sinc=10 to=Camstats.txt

HAVE A LOOK AT YOUR DATA

qview FC21B0007101_11273005958F3G.cub &

WE NOW CREATE A LIST OF ALL THE IMAGES THAT WILL BE USED BY mosrange:

ls *G.cub > range.lis

AND WE EXECUTE mosrange TO CREATE A MAP THAT WILL BE USED TO MAP PROJECTING THE REST OF THE IMGES

mosrange fromlist=range.lis to=equi.map projection=equirectangular

THE MAP IS USED TO PROJECT THE FIRST IMAGE, AND THEN THE REST OF THE IMAGES ARE PROJECTED TO THE FIRST ONE:

cam2map from=FC21B0007101_11273005958F3G.cub to=FC21B0007101_11273005958F3G-CO-EQ.cub map=equi.map pixres=map lonseam=continue

cam2map from=FC21B0007102_11273010009F4G.cub to=FC21B0007102_11273010009F4G-EQ.cub map=FC21B0007101_11273005958F3G-CO-EQ.cub matchmap=true

cam2map from=FC21B0007106_11273010035F8G.cub to=FC21B0007106_11273010035F8G-EQ.cub map=FC21B0007101_11273005958F3G-CO-EQ.cub matchmap=true

COMPARE THE NON-PROJECTED AND THE PROJECTED IMAGES:

qview FC21B0007102_11273010009F4G.cub FC21B0007102_11273010009F4G-EQ.cub &

USING coreg WE ASSURE THAT EACH FILTER IS CORRECTLY ALIGNED

coreg from=FC21B0007102_11273010009F4G-EQ.cub match=FC21B0007101_11273005958F3G-CO-EQ.cub to=FC21B0007102_11273010009F4G-CO.cub deffile=coreg.def onet=FC21B0007102_11273010009F4G.co.net transform=translate interp=bilinear rows=32 columns=32

coreg from=FC21B0007106_11273010035F8G-EQ.cub match=FC21B0007101_11273005958F3G-CO-EQ.cub to=FC21B0007106_11273010035F8G-CO.cub deffile=coreg.def onet=FC21B0007106_11273010035F8G.co.net transform=translate interp=bilinear rows=32 columns=32

photomet PHOTOMETRICALLY CORRECTS EACH IMAGE BASED OFF OF PUBLISHED VALUES FOR EACH FILTER

photomet -batchlist=base.lis from=$(sed -n '3p' base.lis)-CO.cub to=$(sed -n '3p' base.lis)-pho.cub phtname=hapkehen theta=18.1 wh=0.408 hg1=0.233 hg2=1 hh=0.07 b0=1.7 zerob0standard=false normname=albedo incref=30.0 incmat=0.0 thresh=10e30 albedo=1.0 USEDEM=TRUE

photomet -batchlist=base.lis from=$(sed -n '2p' base.lis)-CO.cub to=$(sed -n '2p' base.lis)-pho.cub phtname=hapkehen theta=18.7 wh=0.377 hg1=0.231 hg2=1 hh=0.07 b0=1.7 zerob0standard=false normname=albedo incref=30.0 incmat=0.0 thresh=10e30 albedo=1.0 USEDEM=TRUE

photomet -batchlist=base.lis from=$(sed -n '1p' base.lis)-CO-EQ.cub to=$(sed -n '1p' base.lis)-pho.cub phtname=hapkehen theta=17.6 wh=0.534 hg1=0.222 hg2=1 hh=0.07 b0=1.7 zerob0standard=false normname=albedo incref=30.0 incmat=0.0 thresh=10e30 albedo=1.0 USEDEM=TRUE

WE CREATE A LIST WITH THE PHOTOMETRICALLY CORRECTED IMAGES AND PERFORM SOME DIVISIONS USING algegra:

ls *pho.cub > pho.lis

algebra from=$(sed -n '1p' pho.lis) from2=$(sed -n '3p' pho.lis) to=750over430.cub operator=divide
algebra from=$(sed -n '1p' pho.lis) from2=$(sed -n '2p' pho.lis) to=750over900.cub operator=divide
algebra from=$(sed -n '3p' pho.lis) from2=$(sed -n '1p' pho.lis) to=430over750.cub operator=divide

WE THEN PUT THE RATIO IMAGES INTO ONE SINGLE CUBE AND TRIM THE AREAS THAT DO NOT OVERLAP:

ls *over*.cub > Ratset.lis

cubeit from=Ratset.lis to=Phoset.cub

bandtrim from=Phoset.cub to=Oppia_Ratio.cub