



Fundamentals of Optical Navigation

William M. Owen, Jr.

Jet Propulsion Laboratory,

California Institute of Technology

June 4, 2018

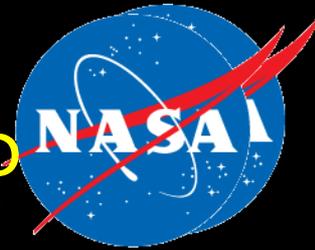


Outline

1. Introduction
2. Geometry (scene prediction)
3. Image processing



Introduction

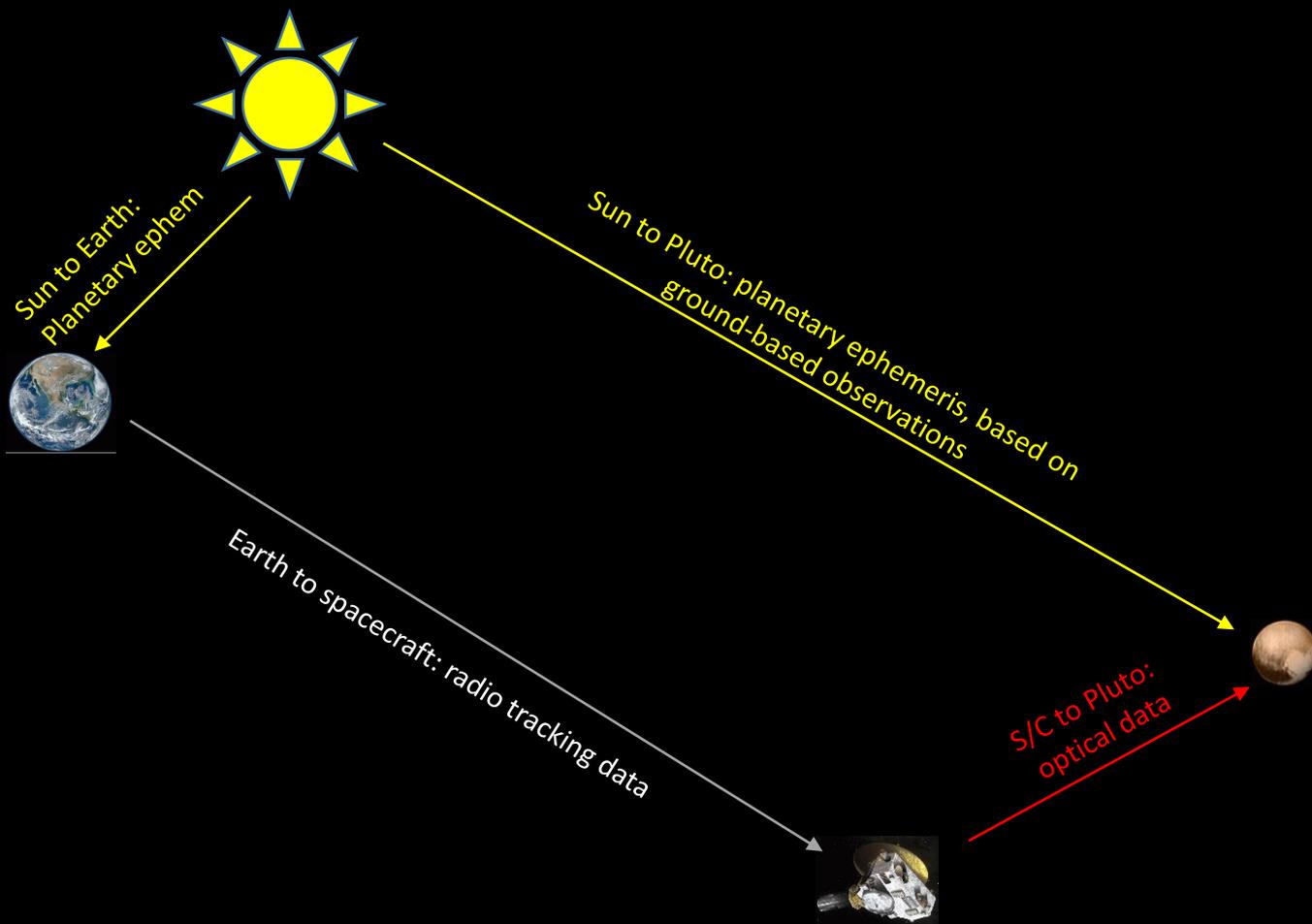


What is Optical Navigation?

- Optical navigation is the use of spacecraft imagery to help determine the trajectory of the spacecraft
- Byproducts:
 - Orbits of the observed bodies
 - Sizes, shapes, rotational parameters
 - Terrain maps (heights, slopes, albedos)
- We care WHERE they are



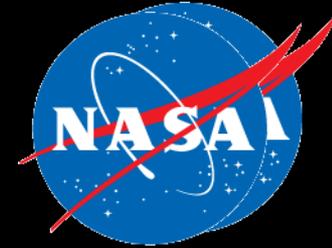
Why optical data?





Terminology

- “Camera” = some sort of imaging device
- “Detector” = light-sensitive thing inside the camera
- “Picture” = what cameras produce
 - “Pixel” = “picture element” = smallest component of a picture
 - “DN” = “Data Number” = digital content of a pixel
- “Object” = what you’re taking a picture of
- “Image” = what pictures contain
- “Projection” = mapping of objects into images



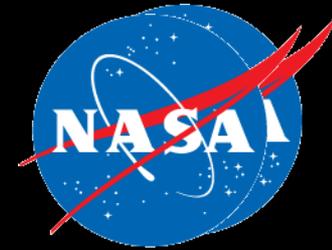
Opnav tasks

1. Size the problem

Work with the rest of the nav team

- What is the accuracy of the camera?
- What is the sensitivity of the camera?
- How many pictures do we need?
- How often?
- What operational constraints?

This is engineering.



Opnav tasks

2. Develop a picture schedule

Work with the science planners and sequencers

- Negotiate observing time with the spacecraft camera(s)
- Find good-looking pictures
 - Lots of stars behind the bodies
 - Good geometry for landmark tracking
- Create sequences of commands to take the pictures
- Verify that these will meet nav requirements

This is geometry.



Opnav tasks

3. Analyze the pictures as they come in

Work again with the rest of the nav team

- Calibrate the camera
- Determine the observed coordinates of
 - Star images
 - Center of body images
 - Landmark images
- Look for outliers, iterate, get appropriate data weights

This is image analysis.

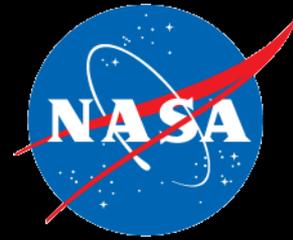


Opnav tasks

4. Reconstruction

- “There’s never enough time to do it right, but there’s always enough time to do it over”
- Lessons learned!
- Papers to write!
- Talks to present!

This is common sense and common courtesy.



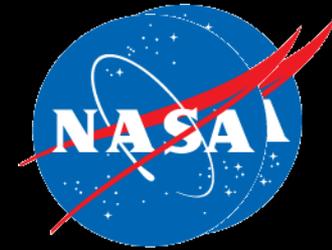
Geometry



Coordinate systems

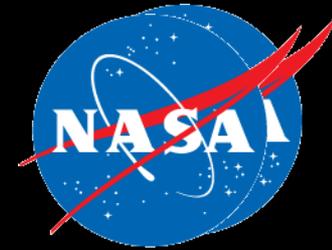
- Specified by an origin and directions of three mutually perpendicular principal axes
- Can be right-handed or left-handed
 - Math works the same either way
- May be rotating
- May be translating

- Opnav really deals with only three



Coordinate systems

- International Celestial Reference System
 - THE fundamental coordinate system
 - Intended to be inertial
 - Origin at barycenter of Solar System
 - X axis toward* the vernal equinox at 2000 Jan 01 12:00
 - Z axis toward* the earth's north pole
 - Right-handed
 - "Earth mean equator and equinox of J2000"
- *At least that was the intent



Coordinate systems

- Camera coordinates
 - Z axis is the optical axis of the system
 - Positive Z points outward, from the camera to the scene
 - +X axis points to the RIGHT in the picture
 - More on that later
 - Right-handed system, so +Y axis must point DOWN in the picture
- How to get from ICRF to Camera is mission specific
 - Just use SPICE! Give it a C kernel, call CKGP, life is good.



Coordinate systems

- Body-fixed coordinates
- +Z axis points to positive angular momentum vector
 - Except for Venus and Uranus (long story)
- Zero longitude is usually defined by some feature on the surface



Vectors

- Magnitude and direction
- Values of their rectangular components depend on the coordinate system in use
- When we talk about “rotating a vector,” we usually mean that the vector stays the same but we’re expressing it in a different coordinate system



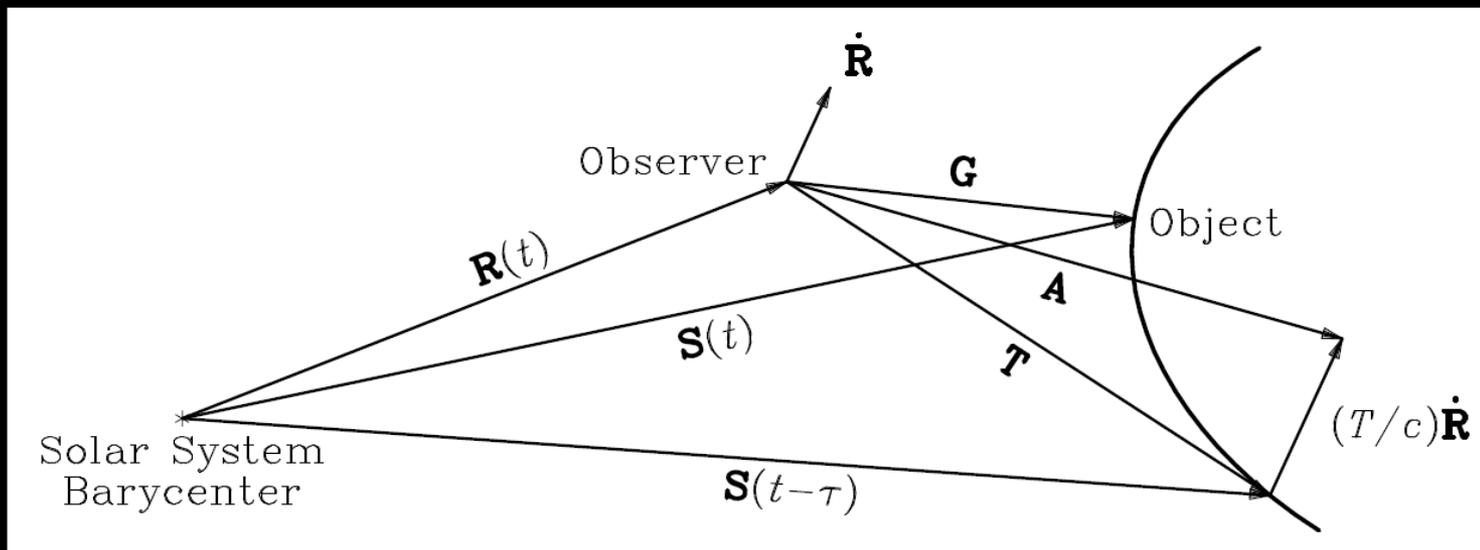
Scene prediction

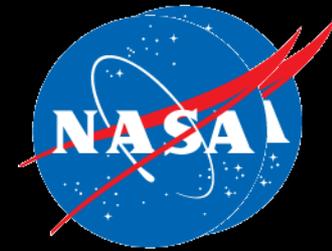
- We need:
 - Position and velocity of camera w.r.t. Solar System Barycenter at the time of the picture: $\mathbf{R}_{\text{cam}}(t)$ and $\mathbf{V}_{\text{cam}}(t)$
 - Apparent position of imaged objects w.r.t. SSB: $\mathbf{A}(t)$
 - Orientation of the camera: “C matrix”
 - Camera’s geometric parameters:
 - Focal length f
 - Number of columns (“samples”) and rows (“lines”) in the detector
 - Pixel dimensions
 - Pixel coordinates of the optical axis
 - Optical distortion parameters



Scene prediction

- “Geometric” position $\mathbf{G}(t) = \mathbf{R}_{\text{obj}}(t) - \mathbf{R}_{\text{cam}}(t)$
- “True” position $\mathbf{T}(t) = \mathbf{R}_{\text{obj}}(t-\tau) - \mathbf{R}_{\text{cam}}(t)$ where τ is the “light time”
 - $\tau = |\mathbf{R}_{\text{obj}}(t-\tau) - \mathbf{R}_{\text{cam}}(t)| / c$
- “Apparent” position $\mathbf{A}(t) = \mathbf{T}(t) + \tau \mathbf{V}_{\text{cam}}(t)$





Scene prediction

- For stars, it's usually OK to use linear propagation for the right ascension and declination:

$$\alpha(t) = \alpha(t_\alpha) + \mu_\alpha (t - t_\alpha)$$

$$\delta(t) = \delta(t_\delta) + \mu_\delta (t - t_\delta)$$

- (α, δ) are the R.A. and Decl.
- (μ_α, μ_δ) are the proper motions (beware a factor $\cos \delta$ in μ_α)
- Star's true or "astrometric" position vector is
$$\mathbf{T}(t) = (1/\pi) (\cos \alpha \cos \delta, \sin \alpha \cos \delta, \sin \delta) \text{ parsecs}$$
 - π is the parallax, not 3.14...
- Rectilinear propagation is rigorous, but you need the star's distance and radial velocity



Gnomonic projection

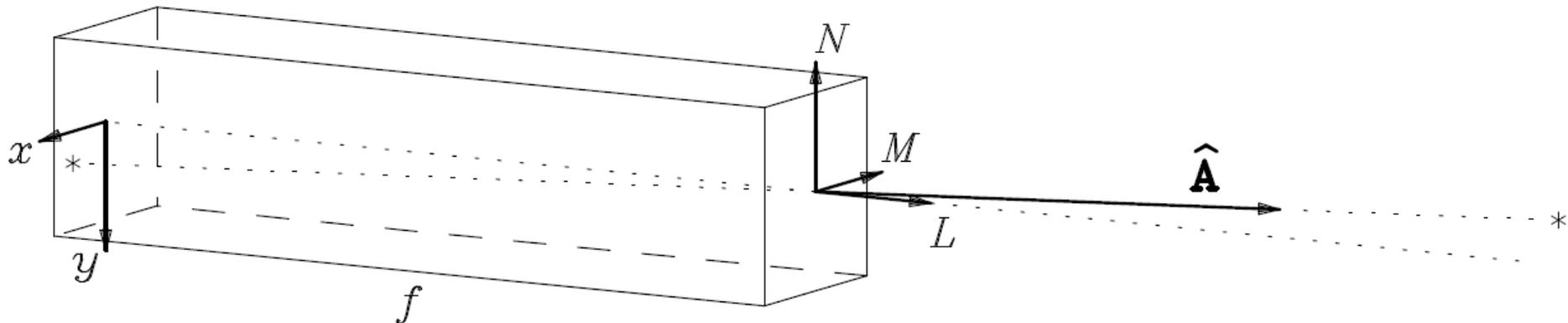
- Yes, just like a sundial!
- Sunlight hits the each point on the gnomon ...
- ... and projects onto one point on a plane

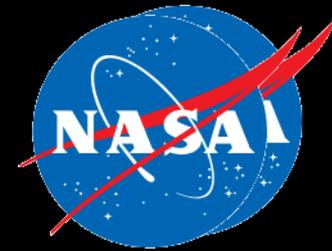




Gnomonic projection

- Cameras work much the same way.
- A light ray from the scene being imaged projects onto one point in the focal plane.
- A pinhole camera follows the gnomonic projection exactly.





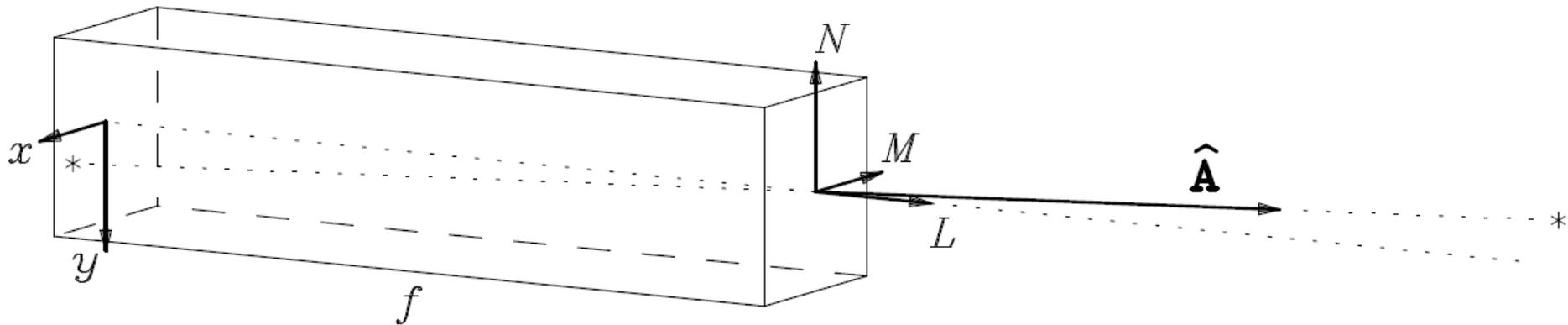
Gnomonic projection

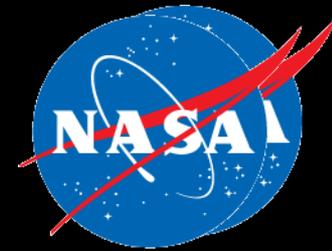
- The math is straightforward:

$$x = f (A_1/A_3)$$

$$y = f (A_2/A_3)$$

once you have rotated \mathbf{A} into camera coordinates





Camera distortions

- Real cameras may not reproduce the gnomonic projection exactly.
- Optical aberrations ($\sin \theta \neq \theta$) and lens or detector misalignments produce these common effects:

- Cubic radial distortion

$$x += \alpha_1 x r^2$$

$$y += \alpha_1 y r^2$$

- “Tip” and “tilt” corrections

$$x += \alpha_2 x y + \alpha_3 x^2$$

$$y += \alpha_2 y^2 + \alpha_3 x y$$



Detector distortions

- You can't always assume that the pixels are square or even rectangular. The mapping from a location (x, y) on the detector into pixel coordinates (s, l) is usually taken to be linear:

$$s = s_0 + K_x x$$

$$l = l_0 + K_y y + K_{yx} x$$

- Here s_0 and l_0 are the pixel coordinates of the optical axis, and the "K matrix" is in pixels/mm (if f is in mm)
- No need for K_{xy} : twist correction can handle it
- Mirror image? No problem. Make K_y negative!



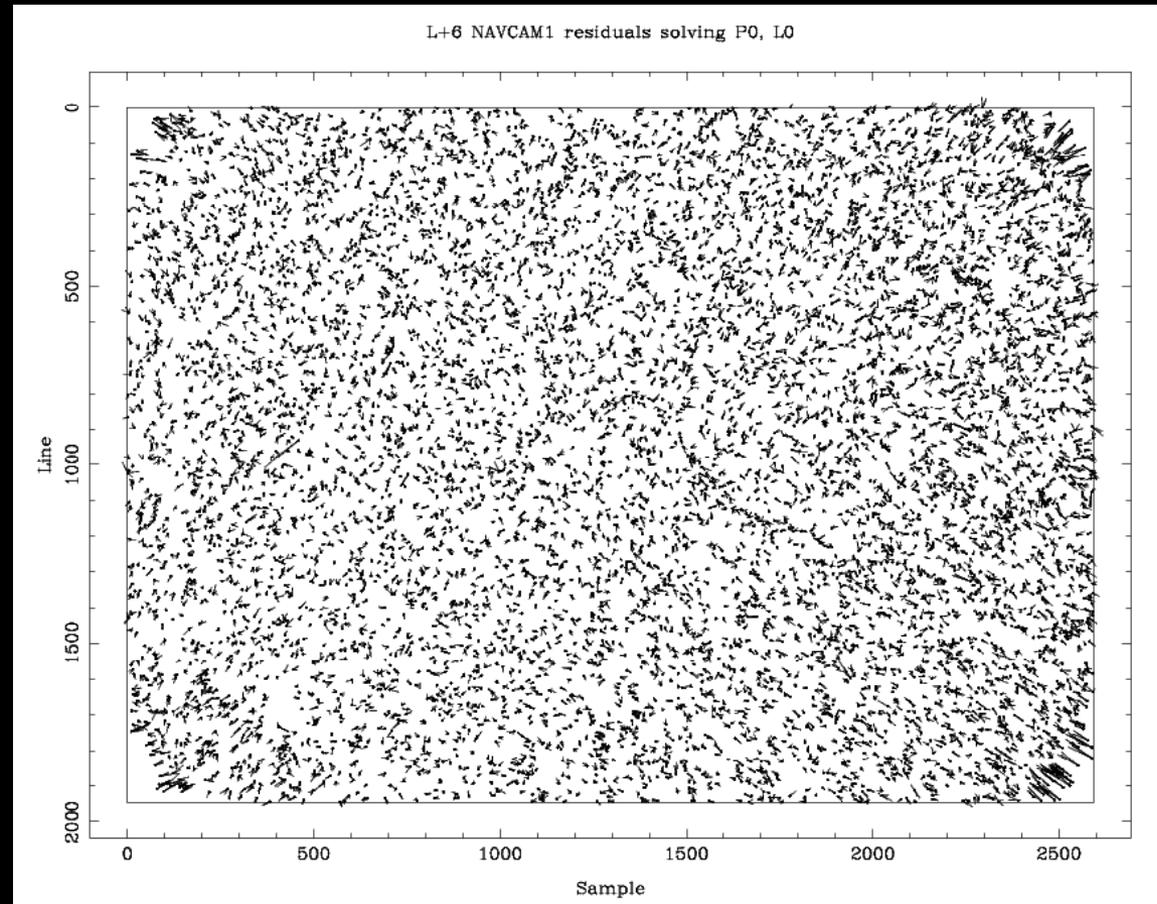
Camera calibration

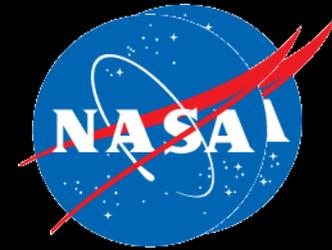
- Best done with pictures of a dense star field
- We observe the product of f and K_x in the horizontal direction, likewise $(f K_y)$ vertically
- Standard practice is to hold K_x fixed at the chip manufacturer's value and solve for f , K_y , and α_i
- Place (s_0, l_0) at the center of the chip unless the distortion pattern is noticeably off-center
- Solve for 3 correction angles to camera orientation too (to update the Frames kernel)



Camera calibration

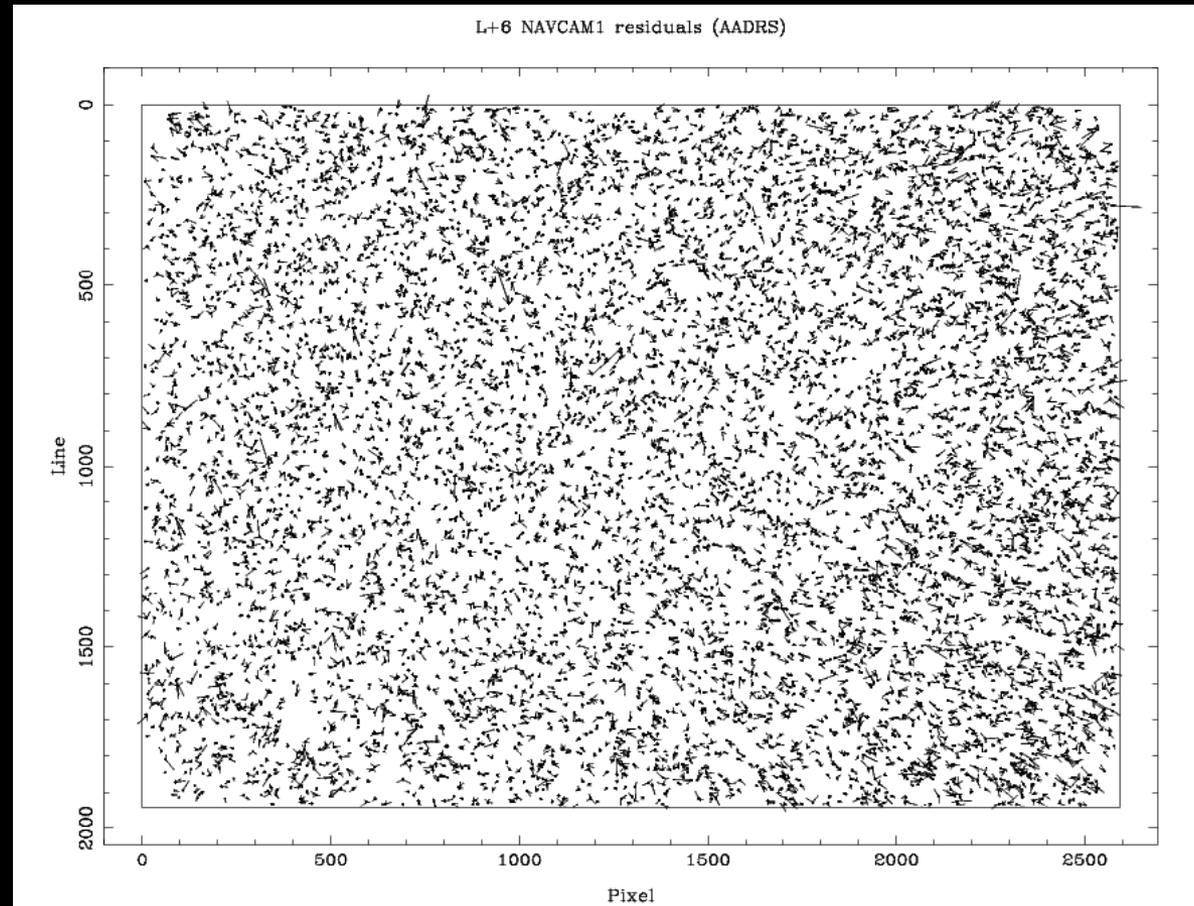
This plot shows postfit residuals for a wide-field camera, solving through 5th order distortion terms. Note that there are still residual trends in the corners.





Camera calibration

After 7th order terms are included in the fit, the corners look ever so much nicer!



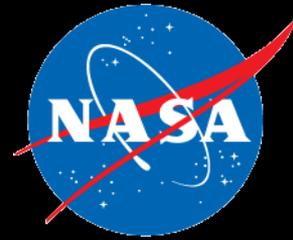


Image Processing



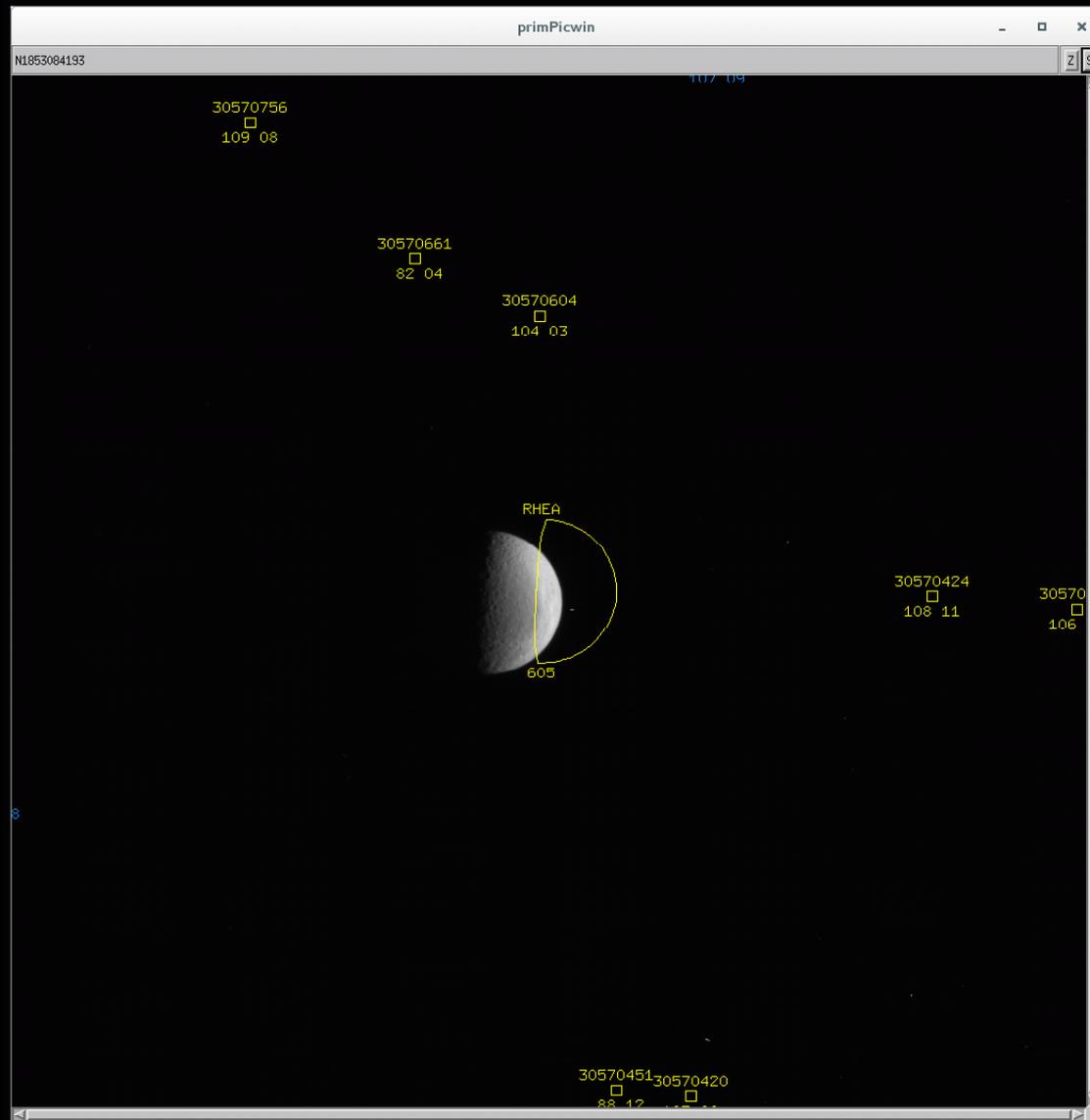
Image Processing

- Everything up to now has dealt with geometry:
 - How to find the apparent R.A. and Decl.
 - How to project that into picture coordinates
- The real trick is how to extract the “observed” coordinates (s, l) from a picture
- Then compare observed (s, l) to predicted, form residuals, calculate partials, filter ...



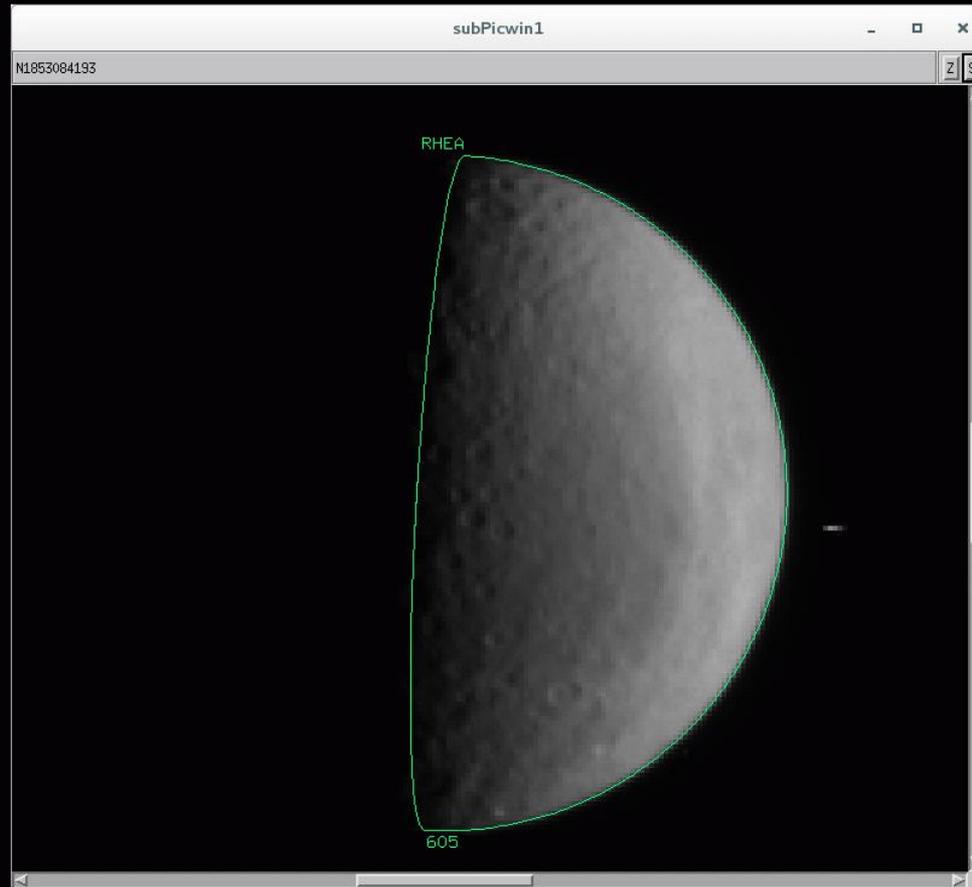
Here's a real example

Cassini picture of
Rhea, Sept. 2016,
1.8 million km



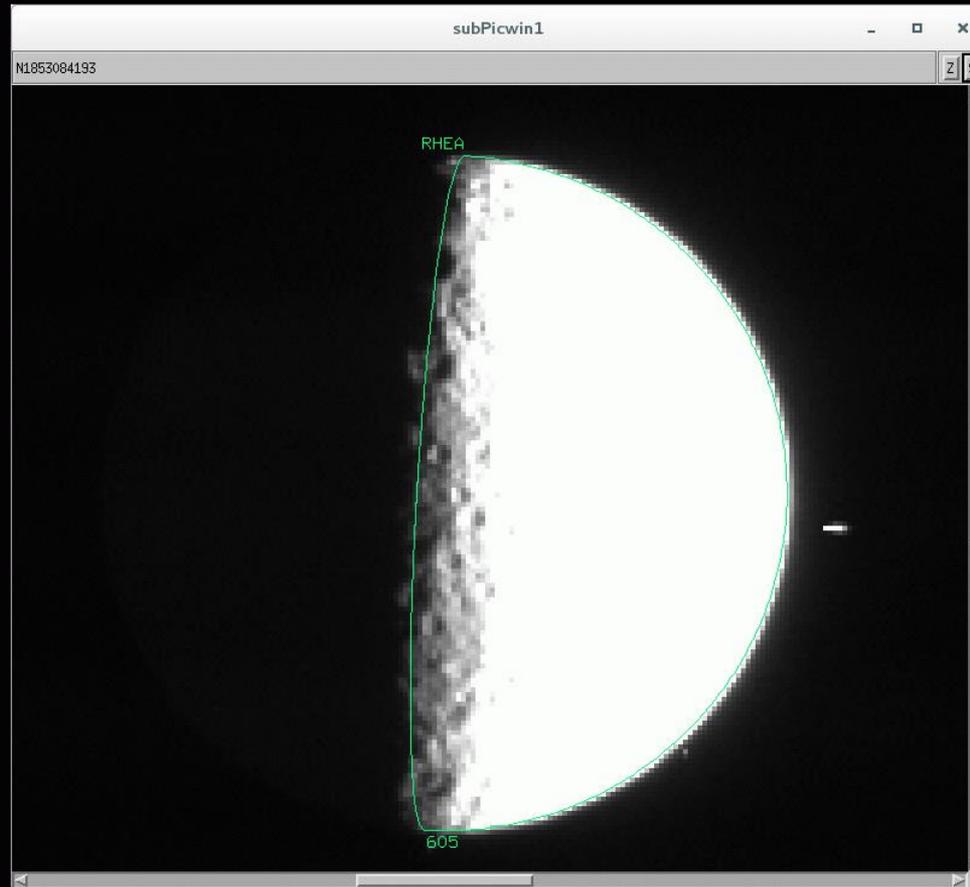


Here's a real example



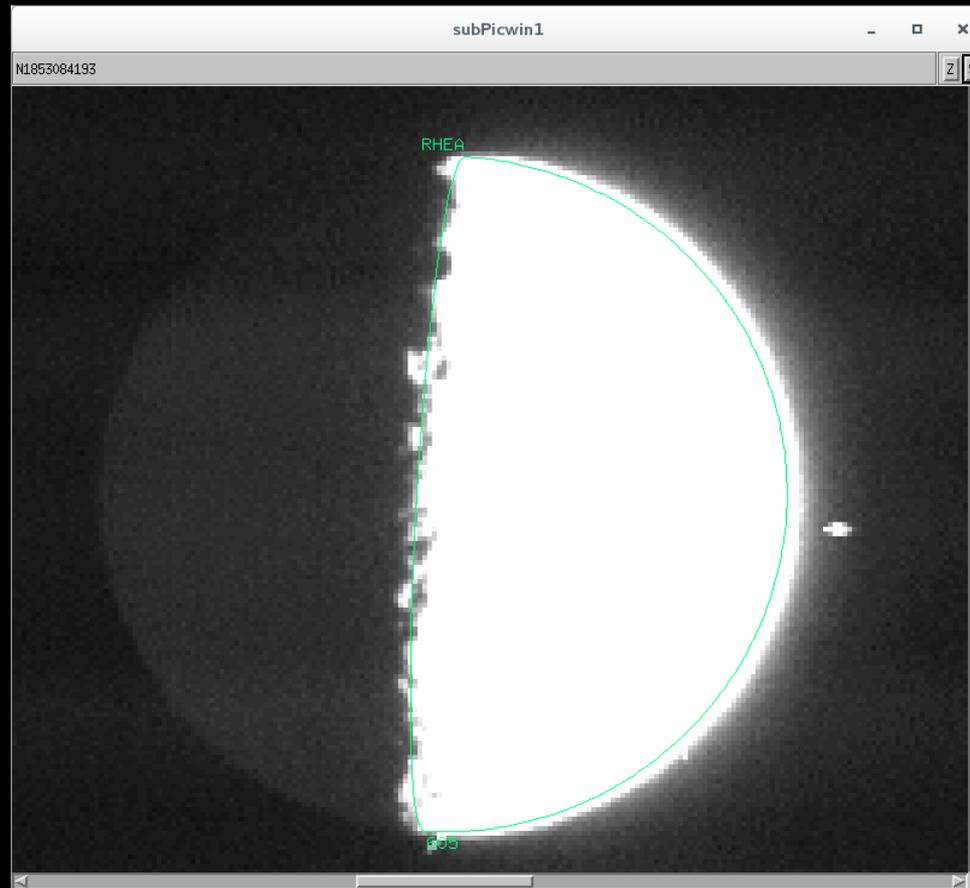


Here's a real example





Here's a real example





A note on displaying pictures

- JPL historically puts the first pixel of a picture file in the UPPER LEFT corner. For all missions through Galileo (and some since) this convention produces a true image of the sky. Just like reading a page.
- Some commercial s/w displays the first pixel in the LOWER LEFT corner.
- The math is the same either way.

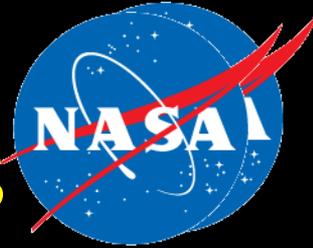


Image Processing Methods

Really there aren't very many:

1. Correlation of the picture against a synthetic (pixelated) image
2. Fitting an analytic function to the pixel data
3. Limb fitting using radial scans



Correlation

- Extract a rectangular portion of the picture that contains the image of the object in question
- Create a rectangular array containing the predicted brightness of the image
- Correlate the two, either by brute force or by Fourier transform
- The (s, l) position of the correlation peak is taken to be the observed location of the image
- SPC uses correlation to find landmarks



Model fit

- Develop an analytic brightness model for the image
- For stars, this might be a circular Gaussian, so the predicted brightness would be

$$A + B \exp \left(-[(x - x_c)^2 + (y - y_c)^2] / \sigma^2 \right)$$

where (x_c, y_c) are the desired image coordinates, σ is the standard deviation of the Gaussian, B is its height, and A is a constant background

- Use standard least squares methods to solve for these five parameters



Model fit

- The preceding is just one example, for stars that could be modeled by a circular Gaussian
- Obviously the same principle applies to other models, even for some irregularly shaped body with variable albedo



Limb scanning

- This has been the preferred method at JPL for finding centers of “extended bodies”
- Construct a set of rays in the picture (half-planes in space) emanating from the assumed center of the image, at various “scan angles”
- Get the observed brightness in the picture along each scan line
- Compute what the expected brightness should be
- Correlate to find the (s, l) position of each limb point



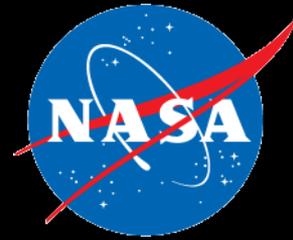
Limb scanning

- Once you have the set of limb points, use least squares to infer the (s, l) position of the center of the body
- Iterate until converged
- Notes:
 - Terminator scans are less useful (gradual drop-off in brightness)
 - Hills and valleys on the limb can influence results



Pointing solution?

- If there are star images in the picture, the camera pointing can be well determined. Perform a pointing solution, and then (maybe?) the nav team can hold the pointing fixed thereafter.
- If there are no stars, then the camera pointing and the spacecraft position are highly correlated. It may make sense to solve for the pointing angles for each picture in the orbit determination filter, with some appropriate *a priori* constraint on the angles.



Reconstruction



You thought it was over

- What could you have done better, or differently?
- Did you have the right inputs?
- Do we have better values for radii, reflectance laws, and so forth?
- What have you learned?
- Would some other method have given better results?
- “There’s never time to do it right, but there’s always time to do it over.”



And with that, I'm finished.
Any questions?