Origins Spectral Interpretation Resource Identification Security-Regolith Explorer (OSIRIS-REx) Project

Stereophotoclinometry for Navigation

Andrew Liounis

October 2015

Revision - 0

Goddard Space Flight Center
Greenbelt, Maryland

Table of Contents

List of Tables

List of Figures

1 Introduction

Throughout its mission, the Origins Spectral Interpretation Resource Identification Security-Regolith Explorer (OSIRIS-REx) spacecraft will rely heavily on various optical measurements to provide navigation updates. During Approach through Orbit A, the optical navigation (OPNAV) measurements will take the form of star based navigation, which provides bearing measurements to the body (in this case the asteroid Bennu). Starting in Orbit A and continuing up through Sample Collection the OPNAV measurements will take the form of surface feature tracking, which provides the capability for a full attitude and position update of the spacecraft (assuming enough surface features can be identified). In this document we will examine the mathematical foundation and theory that the OSIRIS-REx operational software uses to compute surface feature OPNAV measurements.

At its core, surface feature navigation is a simple concept comprised of two main steps. First, a photograph of a body is taken and processed either autonomously or by hand to identify the locations of known surface features from the body in the image. Second, the locations of the identified features in the photo and the known asteroid-fixed locations of the surface features provide 2D-3D point correspondences which can be used to triangulate the location and orientation of the camera which took the photo. This process is shown in Fig. 1.

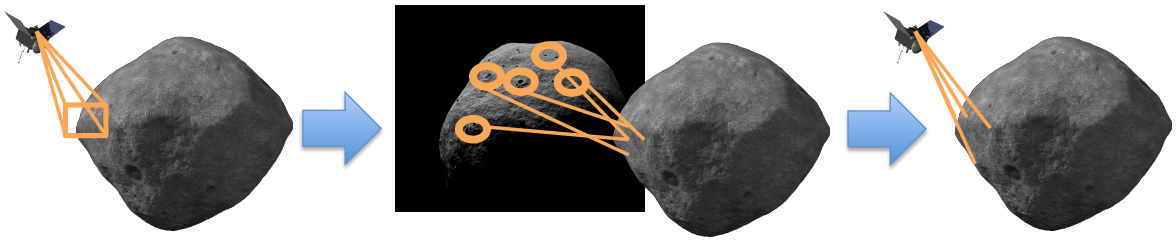


Figure 1: Surface feature navigation is a simple concept. First a photograph is captured of a body with known surface features. Then, the known surface features are located and identified in the photo. Finally the correspondence between the asteroid-fixed location of the known surface features and the location of the known surface features in the photograph frame are used to triangulate the location of the camera that took the photo.

For OSIRIS-REx, these two steps are handled by two different software programs. The identification of surface features is handled autonomously by the Stereophotoclinometry (SPC) programs developed by Dr. Robert Gaskell of the Planetary Sciences Institute. After the surface features have been identified and located in the photos, the point correspondences are passed to the orbit determination software, which updates the spacecraft's positioning and pointing with respect to the body being observed (along with other geodetic parameters) using these correspondences.

In the rest of this paper, we begin by reviewing the pinhole camera model and the numerous frames that will be used throughout the rest of the paper. These sections serve as an introduction to the topics for those who are unfamiliar with them, and as a way to introduce notation to those who are. After the discussion of the camera models and coordinate frames, we continue by examining the surface feature navigation measurement model, which manipulates the 2D-3D point correspondences into updates to the spacecraft state. We use the discussion of the measurement model to both introduce the concept of using 2D-3D point correspondences to navigate and to outline the results we need to get out of the SPC image processing. After our discussion of the measurement model, we turn our attention to the SPC navigation routines that will be used to process the OPNAV photos. We begin this section with a broad overview of the SPC process in general as well as the SPC navigation process, and then proceed to discuss some of the finer implementation details of the SPC navigation process.

1.1 Frames and frame transformations

OPNAV type measurements frequently utilize many different coordinate frames (which is also true for many other space-based navigation methods). The processes involved in surface feature navigation (particularly SPC) are certainly not the exception to this rule. Therefore, in an attempt decrease confusion later, we will now layout all of the different coordinate frames and the steps needed to change between them.[1]

To begin our discussion of coordinate frames, consider the three pictured in Fig. 2. The primary frame is this system is the asteroid-fixed frame. The asteroid-fixed frame is centered at the approximate center of mass of Bennu and rotates with Bennu[2]. The asteroid-fixed frame is key because the locations of the surface features are fixed in time, which is not the case in the inertial frame. The next frame that will be used is the camera frame which is described in detail in Section 1.1.1. The final frame shown in Fig. 2 is the local maplet frame. In fact, there are numerous local maplet frames, one for each maplet that is created by the SPC modelling processes. The maplet frames are all centered at the location of the landmark on the surface of the body. The maplet and the maplet frame will be discussed more thoroughly in Section 3.1.

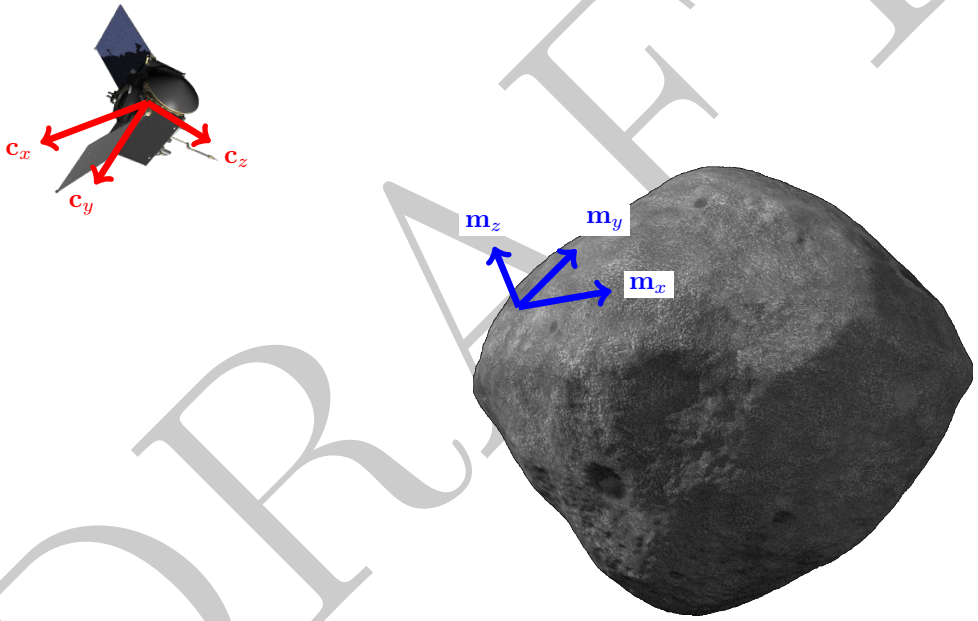


Figure 2: There are three primary frames that we need to worry about (beside the frames used for the pinhole camera model). They are the asteroid-fixed frame (shown in orange), the maplet frame (shown in blue) and the camera frame (shown in red). This figure is not drawn to scale.

The steps required to transform between each frame are relatively simple. It will always require some rotation and translation which we can write as

$$\mathbf{v}_A = \mathbf{T}_A^B(\mathbf{v}_B + \mathbf{a}_B) \tag{1.1}$$

where $\mathbf{v}_A$ is the representation of vector $\mathbf{v}$ in frame A, $\mathbf{v}_B$ is the representation of vector $\mathbf{v}$ in frame B, $\mathbf{T}_A^B$ is the rotation matrix from frame B to frame A, and $\mathbf{a}_B$ is the location of the origin of frame A expressed

[1] Whether or not we are successfully making things less confusing is certainly up for debate. At the very least this section provides a fun opportunity to show off my lack of skills when it comes to drawing coordinate frames in tikz-3dplot and PowerPoint.

[2] In practice, it is almost impossible to determine the true center of mass of the asteroid without a long and detailed observation. Because of this we generally use the center of figure (center of the volume of the asteroid) as an approximate. This leads to two options when it comes to estimating the spacecraft's state. We can either estimate it relative to the center of figure and then estimate the first order gravity harmonics to account for the offset between the center of figure and center of mass, or we can estimate the state relative to the center of mass by estimating an offset between the center of figure and center of mass. Both of these techniques are valid and this document is not focused on the estimation processes for the most

in frame B.[3] In addition, it is easy to find the rotation matrix from one frame to another. If you have the directions of the axes of one frame expressed in another as unit vectors (i.e. you have 3 unit vectors defined in the asteroid-fixed frame that form the axes for the camera frame), say $(\hat{\mathbf{c}}_x)_B$, $(\hat{\mathbf{c}}_y)_B$, and $(\hat{\mathbf{c}}_z)_B$ then the rotation matrix to go from frame C to frame B is simply[4]:

$$\mathbf{T}_B^C = \begin{bmatrix} (\hat{\mathbf{c}}_x)_B & (\hat{\mathbf{c}}_y)_B & (\hat{\mathbf{c}}_z)_B \end{bmatrix} \tag{1.2}$$

where $(\hat{\mathbf{c}}_\bullet)_B$ represents the unit vector in the direction of the $\bullet$ axis of the C frame expressed in the B frame and $\mathbf{T}_B^C$ is the rotation matrix from the C frame to the B frame.

1.1.1 A brief review of camera models

Perhaps the biggest use of frames in SPC for navigation is the pinhole camera model. The pinhole camera model attempts to describe how objects in three-dimensional space are projected onto a two-dimensional plane to form a photograph. It assumes that the world is being viewed through a pinhole, such that light travels in a straight path from the object, through the pinhole, and onto the focal plane of the camera. Thus, the pinhole camera model is actually just a simple gnomic projection from $\mathbb{R}^3$ to $\mathbb{R}^2$. In this section we briefly develop the pinhole camera model, loosely following the description in [1].

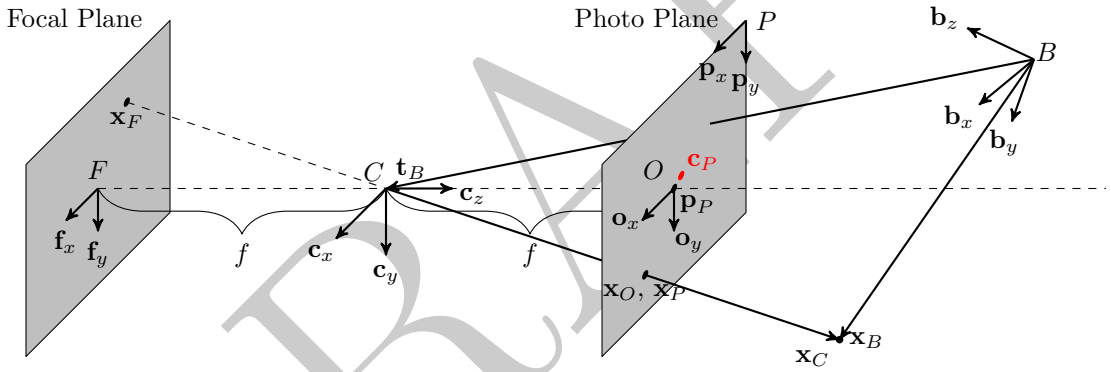


Figure 3: A point, $\mathbf{x}_B$, is projected onto the image plane through the pinhole camera model.

To develop the mathematics behind the pinhole camera model, consider the scene in Fig. 3. In the scene, we have a point, $\mathbf{x}_B$ defined in frame B. We want to project this point onto the focal plane of the camera with focal length f and camera center located at $\mathbf{t}_B$ in frame B. Our first step is to express $\mathbf{x}_B$ in the camera frame (frame C). We can do this using the simple rotation and translation given by

$$\mathbf{x}_C = \mathbf{T}_C^B \left(\mathbf{x}_B - \mathbf{t}_B\right) \tag{1.3}$$

where $\mathbf{x}_C$ is point $\mathbf{x}_B$ expressed in the camera frame, $\mathbf{t}_B$ is the location of the camera center (and origin of the camera frame) in frame B, and $\mathbf{T}_C^B$ is a rotation matrix from frame B to the camera frame. Further, we can also express this transformation from frame B to frame C using homogeneous coordinates as

$$\mathbf{x}_C = \mathbf{T}_C^B \left[\begin{array}{c|c} \mathbf{I}_{3\times 3} & -\mathbf{t}_B \end{array}\right] (\mathbf{x}_h)_B = \mathbf{E}(\mathbf{x}_h)_B \tag{1.4}$$

part; therefore, we will no longer discuss this topic.

[3] Throughout this work we will use the following notation: bold upright uppercase letters will indicate matrices, bold upright lowercase letters will indicate column vectors, and non-bold characters will be scalars. Furthermore, uppercase non-bold subscripts will indicate the frame that a vector is expressed in. Finally, hats over a vector will indicate a unit vector.

[4] I would expect that most people reading this will already know this. In spite of this I have included this in case anyone may not be familiar with this and as a quick reference to myself, as I find that I frequently need to think about how the orientation of the vectors should be in the rotation matrix.

where $\mathbf{I}_{3\times3}$ is the 3×3 identity matrix, $(\mathbf{x}_h)_B$ is the homogeneous version of $\mathbf{x}_B$, and $\mathbf{E} = \mathbf{T}_C^B \left[\begin{array}{c|c} \mathbf{I}_{3\times3} & -\mathbf{t}_B \end{array} \right]$ is the extrinsic camera matrix (thus called because it is entirely dependent on the scene or external parameters).

Now that we have expressed our point of interest in the camera frame we can begin considering the projection. Start by examining the slice of the c_y–c_z plane from Fig. 3 shown in Fig. 4. As should be apparent, we are

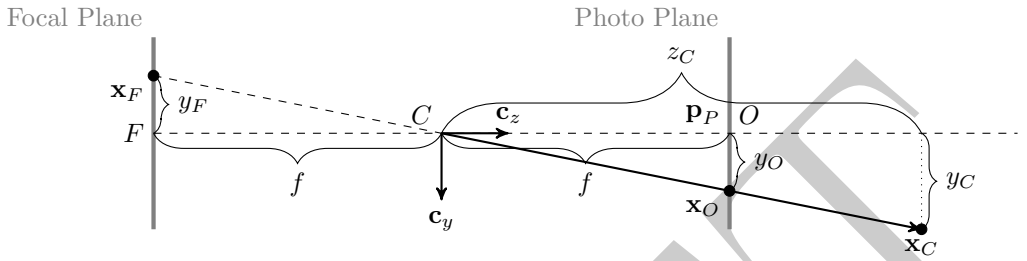


Figure 4: A slice of the c_y–c_z plane from the scene in Fig. 3

.

simply working with similar triangles. To determine the $y-$coordinate of the point in the focal plane, we just need to multiply by the scaling term f/z_c and flip the sign to account for the fact that we have crossed the principal axis. This allows us to express the coordinates where the point projects onto the focal plane as follows:

$$\mathbf{x}_F = -\frac{f}{z_C} \left[\begin{array}{c} x_C \\ y_C \end{array} \right] \tag{1.5}$$

where $\mathbf{x}_F$ is the point expressed in the focal frame and f is the focal length of the camera expressed in units of pixels[5].

This is the simplest version of the pinhole camera model; however, in this case our photo of the real world has actually been flipped upside down and left/right due to the fact that we crossed the principal axis on the way to the focal plane[6]. In most modern cameras, the photo that is output after capturing a scene has been corrected to be in the same orientation as the world and this is what most people expect when they see a photo. In order to account for the internal corrections of the camera it is common to skip using the focal plane projection and instead define a new imaginary "photo plane" placed a distance of f in front of the camera center[7]. Working in the photo frame allows us to work with the photograph as it actually appears (and how scenes appear to our eyes). The only thing we need to change to work in this frame is the negative sign in front of equation 1.5 to give us

$$\mathbf{x}_O = \frac{f}{z_C} \left[\begin{array}{c} x_C \\ y_C \end{array} \right]. \tag{1.6}$$

where $\mathbf{x}_O$ is the location of the point in the photo frame[8].

One assumption that we have made here that is commonly broken is that the principal axis aligns with the origin of the coordinate system we are using in the photo frame. Frequently, the coordinate system origin is

[5]In practice, what we would like to do is take the focal length in units of distance (millimeters, inches, etc.) and then divide by the pixel pitch in order to get the focal length in units of pixels. In practice, however, it is impossible to estimate both the pixel pitch and the focal length of the camera, therefore we need to simply estimate the ratio between them. In the end everything works out the same but it may be confusing to some. In other models, such as the Owen model which will be discussed shortly, we can get around this by assuming a nominal (manufacturer specified) value for the pixel pitch or the focal length, which allows us to continue working in engineering units, but in the end we still do not know the true values for the pixel pitch or the focal length.

[6]As a fun thought experiment: this is also how our eyes work. The "photo" received by our cones and rods is flipped upside down of the orientation of the objects in the real world. Presumably our brain then corrects this image to match the actual orientation of the real world... or does it? Perhaps what we see is actually the inverse of the real world and we would never know it because we have never experienced differently. Of course this might then imply that the photo as it appears on the focal plane would appear in the correct orientation, or would it? Anyway, I digress.

[7]In the literature this is referred to as the "image plane." We have used the word photo here for reasons discussed later.

[8]Note that if the $x-$ and $y-$axes of our picture frame are flipped from the $x-$ and $y-$axes of our photo frame, we can account for this here by changing the sign of the $x-$term, $y-$term, or both terms to account for the flips.

placed in one of the corners of the image, (see frame P in Fig. 3 for an example). In addition, the principal axis may not always be perfectly aligned with the center of the photo plane (note that the principal axis is by definition perpendicular to the photo and focal planes). This then leads to two coordinate systems with origins on our photo plane, the photo frame, whose origin is located at the principal point (frame I in Fig. 3), and the picture frame, whose origin is located at the beginning of the coordinate system used to reference the photo itself (frame P in Fig. 3. To correct for these differences, it is common to include an offset term to account for the difference between the principal point (point $\mathbf{p}$ in Fig. 3) and the origin of the picture coordinates (the picture frame, frame P in Fig. 3). This is given by

$$\mathbf{x}_P = \mathbf{x}_O + \mathbf{p}_P \tag{1.7}$$

where $\mathbf{x}_P$ is the location of the point in the picture frame and $\mathbf{p}_P$ is the location of the principal point in the picture frame. Further, we can describe the entire projection and translation as

$$\mathbf{x}_P = \frac{1}{z_C}\begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \end{bmatrix}\mathbf{x}_C = \frac{1}{z_C}\mathbf{N}\mathbf{x}_C \tag{1.8}$$

where

$$\mathbf{N} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \end{bmatrix} \tag{1.9}$$

is the intrinsic camera matrix[9] (thus called because it is entirely dependent on the camera itself, or the internal parameters) and p_x and p_y are the x and y components of the principal point expressed in the picture frame. Finally, we can define the complete camera matrix to be

$$\mathbf{C} = \mathbf{N}\mathbf{E} \tag{1.10}$$

such that we have

$$\mathbf{x}_P = \frac{1}{z_C}\mathbf{C}(\mathbf{x}_h)_B \tag{1.11}$$

as the full mapping from an arbitrary frame B to the picture frame of a camera. This completes the basic pinhole camera model.

In real life the pinhole camera model does not fully account for everything that is happening in a camera. Things like lens distortions, mis-alignment of the focal plane, and other issues cause differences between the results predicted by the pinhole camera model and the actual results from a camera. Luckily much work has gone into ways to account for these errors through what are called distortion models.

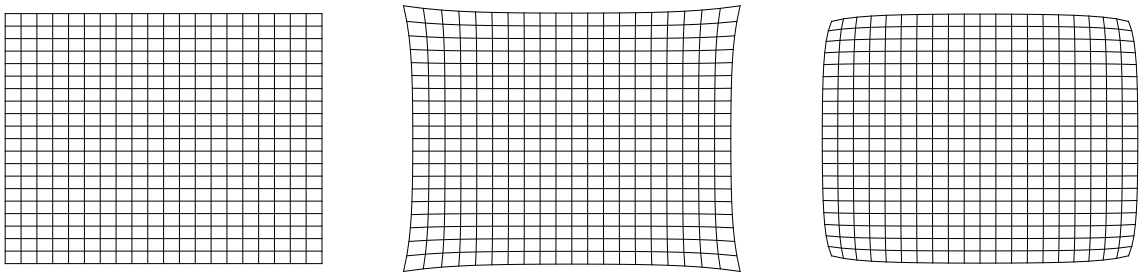
There are numerous distortion models that can be used; however, in this case we will focus primarily on the Owen distortion model because it is used most frequently in SPC processing. The Owen Model, as described in [2] and in the Stereophotoclinometry source code, is very similar to the basic pinhole camera model with a few slight modifications. The first modification is the inclusion of a distortion model to account for real world lenses. In this case, to apply the distortion, we first must transform our point in the camera frame to a point in the photo plane as is done in Eq. 1.6. In this case f is the actual physical focal length of the camera (usually expressed in millimeters). Now that we are in the photo space we can begin applying the distortions.

The first distortion type that is considered by the Owen model is a fourth order radial distortion given by

$$\Delta(\mathbf{x}_O)_{rad} = \epsilon_1 r^2 \mathbf{x}_O + \epsilon_2 r^4 \mathbf{x}_O \tag{1.12}$$

where $\epsilon_{1,2}$ are the radial distortion coefficients and $r = \sqrt{x_O^2 + y_O^2}$ is the radial distance from the principal point of the camera. Radial distortions take the form of pincushion or barrel effects and are caused by the way that lenses bend the light that passes through them. An example of radial distortion is shown in Fig. 5.

[9] In the literature this is matrix $\mathbf{K}$; however, we have switched notation here to avoid confusion with the K matrix in the Owen model.



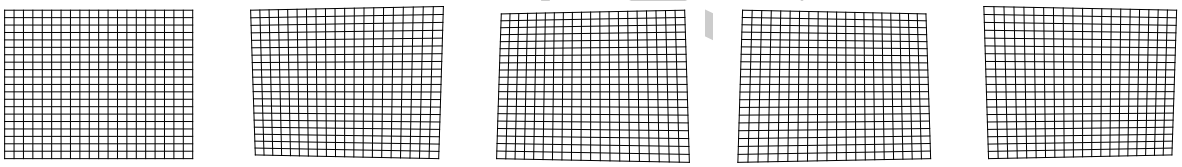
(a) None **(b) Pincushion** **(c) Barrel**

Figure 5: An example of the two different kinds of radial distortion. Pincushion distortion occurs when ϵ_1 and ϵ_2 are positive and barrel distortion occurs when ϵ_1 and ϵ_2 are negative. A mixing of the signs of ϵ_1 and ϵ_2 will lead to a mixing of the distortions.

The next distortion considered is a tangential distortion according to [3] (although it is referred to as a tip/tilt/prism distortion in [2]). This is given by

$$\Delta(\mathbf{x}_O)_{tan} = \epsilon_3 y_O \mathbf{x}_O + \epsilon_4 x_O \mathbf{x}_O \tag{1.13}$$

where $\epsilon_{3,4}$ are the tangential distortion coefficients and x_O and y_O are the coordinates of the point in the photo frame. Tangential distortions are mostly caused by lens elements that are not perfectly aligned with the principal axis [3]. An example of the effect of tangential distortions in shown in Fig. 6.



(a) None **(b)** ϵ_1+, ϵ_2+ **(c)** ϵ_1-, ϵ_2+ **(d)** ϵ_1-, ϵ_2- **(e)** ϵ_1+, ϵ_2-

Figure 6: Examples of tangential distortions. Each subfigure is labeled with the signs of the tangential distortion coefficients that have been used to generate them

The final distortion considered by the Owen model is not documented anywhere and only seems to appear in the SPC version of the Owen model. It appears that it may be some form of radial distortion but the implementation is odd due to the cross with the x and y terms and the negative sign for the x terms. This distortion is given as

$$\Delta(\mathbf{x}_O)_{pin} = (\epsilon_5 r + \epsilon_6 r^3) \begin{bmatrix} -y_O \\ x_O \end{bmatrix} \tag{1.14}$$

where $\epsilon_{5,6}$ are the extra distortion coefficients. This distortion was used to generate the grids in Fig. 7. As can be seen, it appears to create something of a pinwheel[10] effect on the grids. [11]

The total distortion model is found by summing the terms from Eqs. 1.12-1.14:

$$(\mathbf{x}_O)_{dist} = \mathbf{x}_O + \Delta(\mathbf{x}_O)_{rad} + \Delta(\mathbf{x}_O)_{tan} + \Delta(\mathbf{x}_O)_{pin} \tag{1.15}$$

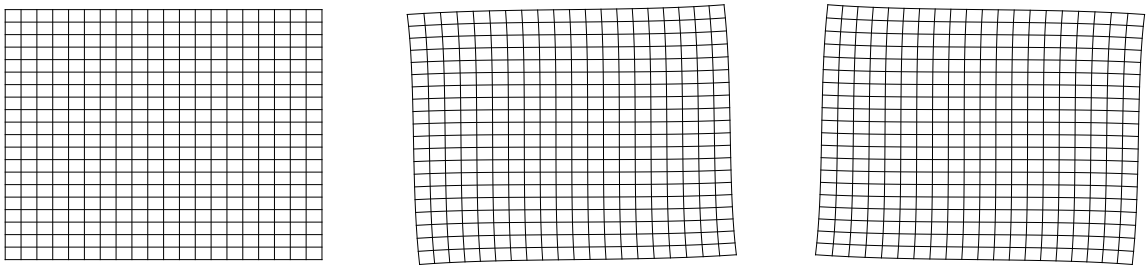
where $(\mathbf{x}_O)_{dist}$ is the distorted location of the point in the photo frame.

Now all that remains is to move the points from the photo frame to the picture frame. This is done using

$$\mathbf{x}_P = \begin{bmatrix} K_x & K_{xy} & p_x \\ K_{yx} & K_y & p_y \end{bmatrix} \begin{bmatrix} (\mathbf{x}_O)_{dist} \\ 1 \end{bmatrix} \tag{1.16}$$

[10] This term does not appear anywhere in the literature. I have come up with it myself

[11] Ask Dr. Gaskell about this!



(a) None **(b) Clockwise** **(c) Counter-clockwise**

Figure 7: Examples of radial pinwheel distortions. Clockwise pinwheels are generated with ϵ_5 and ϵ_6 being positive. Counter-clockwise pinwheels are generated with ϵ_5 and ϵ_6 being negative.

where the K terms scale/rotate into the picture frame and have units of pixels/distance (with distance being the same unit as the focal length)[2]. The K terms in Eq. 1.16 require some more discussion. First, in an ideal detector with square pixels, $K_x = K_y$ and $K_{xy} = K_{yx} = 0$ such that K_x and K_y would just serve as a unit conversion from distance to pixels [2]. If the pixels were rectangular then we would have that $K_x \neq K_y$ while K_{xy} and K_{yx} would still be 0 [2]. It is therefore apparent that K_x and K_y account for both the conversion from distance to pixels as well as potential differences in pixel size (whether by design or not). The off-diagonal terms only come into play if the x and y axes are not perfectly perpendicular due to a manufacturing error and therefore estimate the skewness of the imaging array. In the SPC version of the Owen model there are also two additional terms, K_{xxy} and K_{xyy} which are multiplied by product of the x and y components of the point $(\mathbf{x}_O)_{dist}$ and added to the picture frame coordinates. It is not apparent what these terms are supposed to measure based off of the SPC source code and they are not described in the literature. In addition, in practice it appears that these terms are usually 0, so we will not pay any further attention to them for now. [12]

In practice it is impossible to estimate all of the K parameters due to observability issues, therefore the following is common practice for implementation according to [2]. First, the value of K_x is held fixed at the manufacturer's specified value, while the value of K_y is allowed to vary to account for the non-squareness of the pixels. Second, the value for f is estimated in order to account for changes in overall scale. Finally, the value of K_{xy} is held fixed at 0 while the value of K_{yx} is allowed to vary in order to account for a non-perpendicular angle between the x and y axes of the detector. This completes the Owen camera model.

2 The surface feature navigation measurement model

The end goal for all the processing steps in this paper is to provide information which can be used to update the spacecraft's state. In particular, we are interested in determining updates to the position and attitude of the spacecraft through the use of our 2D-3D point correspondences. In order to do this we need a measurement model and the measurement partials.

Luckily for us, we have already thoroughly examined our measurement model for surface feature navigation. It is simply the camera model (in this case we are using the Owen camera model), which when given the spacecraft's position ($\mathbf{t}_B$) and attitude ($\mathbf{T}_B^C$) as well as the location of the landmark in the asteroid-fixed frame ($\mathbf{x}_B$) provides an estimate of the measurement in the form of the picture frame location of the landmark projected onto the photo plane. We now restate all of the equations needed for the measurement model here

[12] Ask Dr. Gaskell about this!

for convenience. For an in-depth discussion of these equations please refer to the previous section.

$$\mathbf{x}_C = \mathbf{T}_C^B \left(\mathbf{x}_B - \mathbf{t}_B\right) \tag{2.1}$$

$$\mathbf{x}_I = \frac{f}{z_C} \begin{bmatrix} x_C \\ y_C \end{bmatrix} \tag{2.2}$$

$$\Delta(\mathbf{x}_I)_{rad} = \epsilon_1 r^2 \mathbf{x}_I + \epsilon_2 r^4 \mathbf{x}_I \tag{2.3}$$

$$\Delta(\mathbf{x}_I)_{tan} = \epsilon_3 y_I \mathbf{x}_I + \epsilon_4 x_I \mathbf{x}_I \tag{2.4}$$

$$\Delta(\mathbf{x}_I)_{pin} = (\epsilon_5 r + \epsilon_6 r^3) \begin{bmatrix} -y_I \\ x_I \end{bmatrix} \tag{2.5}$$

$$(\mathbf{x}_I)_{dist} = \mathbf{x}_I + \Delta(\mathbf{x}_I)_{rad} + \Delta(\mathbf{x}_I)_{tan} + \Delta(\mathbf{x}_I)_{pin} \tag{2.6}$$

$$\mathbf{x}_P = \begin{bmatrix} K_x & K_{xy} & p_x \\ K_{yx} & K_y & p_y \end{bmatrix} \begin{bmatrix} (\mathbf{x}_I)_{dist} \\ 1 \end{bmatrix} \tag{2.7}$$

With our measurement model in hand all we need is a way to update our state given a measurement and our current best estimate of our state. We can do this in multiple ways, but perhaps the easiest to understand is through the use of a least squares problem.

In our set up, we are looking to determine the state vector that minimizes the distance between all of our predicted and actual measurements. That is

$$\min_{\boldsymbol{\chi}} \sum_{i=1}^{n} \left(\mathbf{f}(\boldsymbol{\chi}, \mathbf{x}_{Bi}) - \mathbf{y}_i\right)^2 \tag{2.8}$$

where $\boldsymbol{\chi} = \begin{bmatrix} \mathbf{t}_B^T & \boldsymbol{\delta\theta}^T \end{bmatrix}^T$ is the state vector, $\boldsymbol{\delta\theta}$ is an additive update to our attitude matrix,[13] $\mathbf{y}_i$ is the i^{th} measurement (picture frame location of the i^{th} landmark), $\mathbf{f}(\boldsymbol{\chi}, \mathbf{x}_{Bi})$ is a vector function of the measurement model (Owen camera model) evaluated using the state vector and the i^{th} landmark location in the asteroid-fixed frame, and we have a total of n landmarks that we have identified in our photo. Note that we may also what to include the asteroid-fixed locations of our landmarks in the state vector as well in which case $\boldsymbol{\chi} = \begin{bmatrix} \mathbf{t}_B^T & \boldsymbol{\delta\theta}^T & \mathbf{x}_{B1}^T & \dots & \mathbf{x}_{Bn}^T \end{bmatrix}$.[14] Regardless of what our state vector is we can equivalently express Eq. 2.8 using matrix-vector notation as

$$\min_{\boldsymbol{\chi}} = \text{Tr}\left[\left(\mathbf{F}(\boldsymbol{\chi}) - \mathbf{Y}\right)^T \left(\mathbf{F}(\boldsymbol{\chi}) - \mathbf{Y}\right)\right] \tag{2.9}$$

where $\mathbf{F}(\boldsymbol{\chi}) = \begin{bmatrix} \mathbf{f}(\boldsymbol{\chi}, \mathbf{x}_{B1}) & \mathbf{f}(\boldsymbol{\chi}, \mathbf{x}_{B2}) & \dots & \mathbf{f}(\boldsymbol{\chi}, \mathbf{x}_{Bn}) \end{bmatrix}$ is a vector which stacks all of the estimated locations of the landmarks into columns, $\mathbf{Y} = \begin{bmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \dots & \mathbf{y}_n \end{bmatrix}$ is a vector which stacks all of the measured locations of the landmarks in the picture frame into columns, and $\text{Tr}[\bullet]$ is the matrix trace operator which sums all of the elements along the diagonal of a matrix.

Now, since we know we are looking for the $\boldsymbol{\chi}$ that minimizes our objective function we want to find the state where the derivative of the objective function is zero[15]. Using the rules of matrix-vector calculus and the properties of the trace operator (namely that $\text{Tr}\left[\mathbf{A}^T\mathbf{B}\right] = \text{vec}(\mathbf{A})^T\text{vec}(\mathbf{B})$) we can write

$$\frac{\partial}{\partial \boldsymbol{\chi}} \left\{\text{vec}\left(\mathbf{F}(\boldsymbol{\chi}) - \mathbf{Y}\right)^T \text{vec}\left(\mathbf{F}(\boldsymbol{\chi}) - \mathbf{Y}\right)\right\} = \mathbf{0}_{1\times 6} \tag{2.10}$$

[13] This means that we can represent our true attitude as $\left(\mathbf{T}_C^B\right)^+ = \left(\mathbf{I}_{3\times 3} - [\boldsymbol{\delta\theta}\times]\right)\left(\mathbf{T}_C^B\right)^-$ where a superscript of $+$ indicates the updated attitude matrix, a superscript of $-$ indicates the previous attitude matrix, and $[\bullet\times]$ indicates a skew symmetric matrix formed from the vector $\bullet$.

[14] If we wish to do this, however, we must be careful to ensure that we have enough information to make all of the states observable, so it may be better to make the asteroid-fixed locations of the landmarks consider parameters instead. This topic in filter practice is unfortunately outside the realm of this paper, so we will leave this topic here.

[15] In the scalar case it is easy to show that the location where the first derivative of this objective function is zero is a minimum since our objective function is quadratic. In this case, where we are not dealing with scalars it is much harder to show this, and in fact in some problems there are multiple locations where the first derivative is equal to zero, some of which may even correspond to local maxima. A discussion on the assurances that we are finding a local minima are outside of the bounds of this paper; however, so we will just proceed and hope that the result we get is truly the minimum (In practice it will be except for some very specific cases which should not happen using real life data).

where $\frac{\partial}{\partial \boldsymbol{\chi}}\{\bullet\}$ indicates the partial derivative of $\bullet$ with respect to $\boldsymbol{\chi}$, $\mathbf{0}_{1\times 6}$ is a row vector with six columns of zeros, and vec($\bullet$) stacks the columns of matrix $\bullet$ into a single column vector (i.e. $\text{vec}\left(\begin{bmatrix}\mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3\end{bmatrix}\right) = \begin{bmatrix}\mathbf{a}_1^T & \mathbf{a}_2^T & \mathbf{a}_3^T\end{bmatrix}^T$). Using the rules of matrix-vector calculus and realizing that $\mathbf{Y}$ is not dependent on $\boldsymbol{\chi}$ we can express this derivative as

$$2\text{vec}\left(\mathbf{F}(\boldsymbol{\chi})-\mathbf{Y}\right)^T \frac{\partial}{\partial \boldsymbol{\chi}}\left\{\mathbf{F}(\boldsymbol{\chi})\right\} = \mathbf{0}_{1\times 6}. \tag{2.11}$$

Now, our goal here is to solve for the $\boldsymbol{\chi}$ that makes this equation true. This is nearly impossible (or actually impossible, I have not checked) to do analytically using the fully non-linear version of $\mathbf{F}(\boldsymbol{\chi})$. Therefore at this point we need to make a linearization using a first order Taylor series expansion about our current best estimate of $\boldsymbol{\chi}$. To do this we use the following substitution

$$\mathbf{f}(\boldsymbol{\chi}^+, \mathbf{x}_{Bi}) \approx \mathbf{f}(\boldsymbol{\chi}^-) + \left.\frac{\partial \mathbf{f}(\boldsymbol{\chi}, \mathbf{x}_{Bi})}{\partial \boldsymbol{\chi}}\right|_{\boldsymbol{\chi}=\boldsymbol{\chi}^-} \left(\boldsymbol{\chi}^+ - \boldsymbol{\chi}^-\right) \tag{2.12}$$

where a superscript of $-$ indicates the CBE of the state vector, a superscript of $+$ indicates the updated state vector based on the measurements, and $\bullet(x)|_{x=y}$ indicates that the function is to be evaluated at y. In addition, in order to simplify our notation we can define

$$\mathbf{H}_i = \left.\frac{\partial \mathbf{f}(\boldsymbol{\chi}, \mathbf{x}_{Bi})}{\partial \boldsymbol{\chi}}\right|_{\boldsymbol{\chi}=\boldsymbol{\chi}^-} = \left[\begin{array}{cc} \frac{\partial \mathbf{x}_{Pi}}{\partial \mathbf{t}_B} & \frac{\partial \mathbf{x}_{Pi}}{\partial \boldsymbol{\delta\theta}} \end{array}\right]_{\boldsymbol{\chi}=\boldsymbol{\chi}^-} \tag{2.13}$$

where $\mathbf{H}_i$ is our i^{th} Jacobian matrix of our measurement model with respect to our state vector evaluated at the current best estimate of the state vector.[16]

Substituting in our linearization assumption and switching to the new notation we can now rewrite Eq. 2.11 as

$$2\text{vec}\left(\mathbf{F}(\boldsymbol{\chi}^+)-\mathbf{Y}\right)^T \frac{\partial}{\partial \boldsymbol{\chi}^+}\left\{\begin{bmatrix} \mathbf{f}(\boldsymbol{\chi}^-, \mathbf{x}_{B1}) + \mathbf{H}_1\left(\boldsymbol{\chi}^+ - \boldsymbol{\chi}^-\right) \\ \mathbf{f}(\boldsymbol{\chi}^-, \mathbf{x}_{B2}) + \mathbf{H}_2\left(\boldsymbol{\chi}^+ - \boldsymbol{\chi}^-\right) \\ \vdots \\ \mathbf{f}(\boldsymbol{\chi}^-, \mathbf{x}_{Bn}) + \mathbf{H}_n\left(\boldsymbol{\chi}^+ - \boldsymbol{\chi}^-\right) \end{bmatrix}\right\} = \mathbf{0}_{1\times 6} \tag{2.14}$$

which is nice because the only dependency on $\boldsymbol{\chi}^+$ is when it appears by itself (within the partial derivative). Therefore we can distribute our partial derivative and get to

$$2\text{vec}\left(\mathbf{F}(\boldsymbol{\chi}^+)-\mathbf{Y}\right)^T \begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \\ \vdots \\ \mathbf{H}_n \end{bmatrix} = \mathbf{0}_{1\times 6}. \tag{2.15}$$

Now in Eq. 2.15 it is not really clear how we can solve for our updated state vector. Therefore, let us rewrite this equation as

$$\begin{bmatrix}\mathbf{H}_1^T & \mathbf{H}_2^T & \dots & \mathbf{H}_n^T\end{bmatrix} \begin{bmatrix} \mathbf{f}(\boldsymbol{\chi}^{-1}, \mathbf{x}_{B1}) + \mathbf{H}_1\left(\boldsymbol{\chi}^+ - \boldsymbol{\chi}^-\right) - \mathbf{y}_1 \\ \mathbf{f}(\boldsymbol{\chi}^{-1}, \mathbf{x}_{B2}) + \mathbf{H}_2\left(\boldsymbol{\chi}^+ - \boldsymbol{\chi}^-\right) - \mathbf{y}_2 \\ \vdots \\ \mathbf{f}(\boldsymbol{\chi}^{-1}, \mathbf{x}_{Bn}) + \mathbf{H}_n\left(\boldsymbol{\chi}^+ - \boldsymbol{\chi}^-\right) - \mathbf{y}_n \end{bmatrix} = \mathbf{0}_{6\times 1} \tag{2.16}$$

where we have substituted in the Taylor series expansion for $\mathbf{F}(\boldsymbol{\chi}^+)$, applied the vec operator, and taken the transpose of the equation. Now we can carry out the inner product like matrix multiplication (not a true inner product since we are multiplying a matrix by a vector, it just looks like one because of our notation)

[16] If we have defined our state vector to include the asteroid-fixed landmark locations then we would expand our Jacobian matrix with n terms of $\frac{\partial \mathbf{x}_{Pi}}{\partial \mathbf{x}_{Bj}}$ which will be equal to 0 except when $i = j$.

from which we get

$$\sum_{i=1}^{n}\left[\mathbf{H}_i^T\left(\mathbf{f}(\boldsymbol{\chi}^-,\mathbf{x}_{Bi})+\mathbf{H}_i\left(\boldsymbol{\chi}^+-\boldsymbol{\chi}^-\right)-\mathbf{y}_i\right)\right]=\mathbf{0}_{6\times1} \tag{2.17}$$

which is simply a set of six equations with six unknowns. If we collect the unknown terms on the left side then we are left with

$$\sum_{i=1}^{n}\left(\mathbf{H}_i^T\mathbf{H}_i\right)\left(\boldsymbol{\chi}^+-\boldsymbol{\chi}^-\right)=-\sum_{i=1}^{n}\left[\mathbf{H}_i^T\left(\mathbf{f}(\boldsymbol{\chi}^-,\mathbf{x}_{Bi})-\mathbf{y}_i\right)\right] \tag{2.18}$$

which are the normal equations for this particular problem. If we define

$$\boldsymbol{\Delta\chi}=\boldsymbol{\chi}^+-\boldsymbol{\chi}^-$$

$$\boldsymbol{\Delta Y}=\begin{bmatrix}\left(\mathbf{y}_1-\mathbf{f}(\boldsymbol{\chi}^-,\mathbf{x}_{B1})\right)^T & \left(\mathbf{y}_2-\mathbf{f}(\boldsymbol{\chi}^-,\mathbf{x}_{B2})\right)^T & \dots & \left(\mathbf{y}_n-\mathbf{f}(\boldsymbol{\chi}^-,\mathbf{x}_{Bn})\right)^T\end{bmatrix}^T$$

$$\mathbf{H}=\begin{bmatrix}\mathbf{H}_1^T & \mathbf{H}_2^T & \dots & \mathbf{H}_n^T\end{bmatrix}^T$$

then we can write this in the more familiar form of

$$\mathbf{H}^T\mathbf{H}\boldsymbol{\Delta\chi}=\mathbf{H}^T\boldsymbol{\Delta Y}. \tag{2.19}$$

Now that we have the foundation for how we can update our state, we just need to finish defining our Jacobian matrices. This is done by applying simple matrix-vector calculus to Eqs. 2.1-2.7. Further, to make things easier, we can use the chain rule so that we can consider each equation individually. Using the chain rule we have:

$$\frac{\partial\mathbf{x}_{Pi}}{\partial\mathbf{t}_B}=\frac{\partial\mathbf{x}_{Pi}}{\partial\left(\mathbf{x}_{Ii}\right)_{dist}}\frac{\partial\left(\mathbf{x}_{Ii}\right)_{dist}}{\partial\mathbf{x}_{Ii}}\frac{\partial\mathbf{x}_{Ii}}{\partial\mathbf{x}_{Ci}}\frac{\partial\mathbf{x}_{Ci}}{\partial\mathbf{t}_B} \tag{2.20}$$

$$\frac{\partial\mathbf{x}_{Pi}}{\partial\boldsymbol{\delta\theta}}=\frac{\partial\mathbf{x}_{Pi}}{\partial\left(\mathbf{x}_{Ii}\right)_{dist}}\frac{\partial\left(\mathbf{x}_{Ii}\right)_{dist}}{\partial\mathbf{x}_{Ii}}\frac{\partial\mathbf{x}_{Ii}}{\partial\mathbf{x}_{Ci}}\frac{\partial\mathbf{x}_{Ci}}{\partial\boldsymbol{\delta\theta}} \tag{2.21}$$

where $\frac{\partial\mathbf{x}_{Pi}}{\partial\mathbf{t}_B}$ and $\frac{\partial\mathbf{x}_{Pi}}{\partial\boldsymbol{\delta\theta}}$ are the components of the i^{th} Jacobian matrix of the measurement with respect to the state. Further, it should be noted that the first three partial derivatives for these terms are the same, therefore we only need to calculate $\frac{\partial\mathbf{x}_{Pi}}{\partial\mathbf{x}_{Ci}}$ once for each measurement and then right multiply by either $\frac{\partial\mathbf{x}_{Ci}}{\partial\mathbf{t}_B}$ or $\frac{\partial\mathbf{x}_{Ci}}{\partial\boldsymbol{\delta\theta}}$ to get the terms for our Jacobian, where $\frac{\partial\mathbf{x}_{Pi}}{\partial\mathbf{x}_{Ci}}$ is the product of the first three partial derivatives for the i^{th} measurement.[17]

Getting the first three partials is a straightforward application of the rules of matrix-vector calculus, thus we just present the results here without bothering with the "derivation" (The Matrix Cookbook is your friend...):

$$\frac{\partial\mathbf{x}_{Pi}}{\partial\left(\mathbf{x}_{Ii}\right)_{dist}}=\mathbf{K}\begin{bmatrix}\mathbf{I}_{2\times2}\\\mathbf{0}_{1\times2}\end{bmatrix} \tag{2.22}$$

$$\begin{aligned}\frac{\partial\left(\mathbf{x}_{Ii}\right)_{dist}}{\partial\mathbf{x}_{Ii}}=&(1+\epsilon_1r_i^2+\epsilon_2r_i^4+\epsilon_3y_{Ii}+\epsilon_4x_{Ii})\mathbf{I}_{2\times2}+\mathbf{x}_{Ii}\left(2\epsilon_1r_i\frac{\partial r_i}{\partial\mathbf{x}_{Ii}}+4\epsilon_2r_i^3\frac{\partial r_i}{\partial\mathbf{x}_{Ii}}+\begin{bmatrix}\epsilon_4 & \epsilon_3\end{bmatrix}\right)\\&+(\epsilon_5r_i+\epsilon_6r_i^3)\begin{bmatrix}0 & -1\\1 & 0\end{bmatrix}+\begin{bmatrix}-y_{Ii}\\x_{Ii}\end{bmatrix}\left(\epsilon_5\frac{\partial r_i}{\partial\mathbf{x}_{Ii}}+3\epsilon_6r_i^2\frac{\partial r_i}{\partial\mathbf{x}_{Ii}}\right)\end{aligned} \tag{2.23}$$

$$\frac{\partial\mathbf{x}_{Ii}}{\partial\mathbf{x}_{Ci}}=\frac{f}{z_{Ci}^2}\begin{bmatrix}z_{Ci} & 0 & -1\\0 & z_{Ci} & -1\end{bmatrix} \tag{2.24}$$

where $\mathbf{K}$ is the intrinsic camera matrix for the Owen model and

$$\frac{\partial r_i}{\partial\mathbf{x}_{Ii}}=\frac{\mathbf{x}_{Ii}^T}{\sqrt{\mathbf{x}_{Ii}^T\mathbf{x}_{Ii}}} \tag{2.25}$$

[17] If we have defined our state to include the asteroid-fixed landmark locations then we would also have a term where we multiply the first three partials on the right by $\frac{\partial\mathbf{x}_{Pi}}{\partial\mathbf{x}_{Bi}}$ and then a bunch of terms right multiplied by a zero matrix.

is the change in the radial distance from the origin of the photo frame with respect to a change in the points location in the photo frame.

Finding the partials of the camera frame landmark location with respect to the state vector is a little trickier, at least for the attitude component. For the spacecraft asteroid-fixed position component the partial is simply

$$\frac{\partial \mathbf{x}_{Ci}}{\partial \mathbf{t}_B} = -\left(\mathbf{T}_B^C\right)^- \tag{2.26}$$

which is found using standard matrix-vector calculus. For the attitude portion first rewrite Eq. 2.1 as

$$\mathbf{x}_{Ci} = \left(\mathbf{T}_B^C\right)^- (\mathbf{x}_{Bi} - \mathbf{t}_B) - [\boldsymbol{\delta\theta}\times]\left(\mathbf{T}_B^C\right)^- (\mathbf{x}_{Bi} - \mathbf{t}_B) \tag{2.27}$$

where $[\bullet\times]$ indicates a skew-symmetric cross product matrix formed from $\bullet$ and $\left(\mathbf{T}_C^B\right)^-$ is the current best estimate of the camera's attitude with respect to the asteroid-fixed frame. From here we can take the partial derivative with respect to $\boldsymbol{\delta\theta}$

$$\frac{\partial \mathbf{x}_{Ci}}{\partial \boldsymbol{\delta\theta}} = \mathbf{0}_{3\times 3} - \frac{\partial}{\partial \boldsymbol{\delta\theta}}\left\{[\boldsymbol{\delta\theta}\times]\left(\mathbf{T}_B^C\right)^- (\mathbf{x}_{Bi} - \mathbf{t}_B)\right\}. \tag{2.28}$$

Now, what we are trying to take the partial derivative of is a vector; therefore we can equivalently write

$$\frac{\partial \mathbf{x}_{Ci}}{\partial \boldsymbol{\delta\theta}} = -\frac{\partial}{\partial \boldsymbol{\delta\theta}}\left\{\mathrm{vec}\left([\boldsymbol{\delta\theta}\times]\left(\mathbf{T}_B^C\right)^- (\mathbf{x}_{Bi} - \mathbf{t}_B)\right)\right\} \tag{2.29}$$

where $\mathrm{vec}(\bullet)$ is the operator that stacks the columns of a matrix into a single column vector. Performing this step is important because it allows us to write

$$\frac{\partial \mathbf{x}_{Ci}}{\partial \boldsymbol{\delta\theta}} = -\frac{\partial}{\partial \boldsymbol{\delta\theta}}\left\{\left(\left(\mathbf{x}_{Bi}^T - \mathbf{t}_B^T\right)\left(\mathbf{T}_C^B\right)^- \otimes \mathbf{I}_{3\times 3}\right)\mathrm{vec}\left([\boldsymbol{\delta\theta}\times]\right)\right\} \tag{2.30}$$

using the properties of vectorization, where $\otimes$ indicates the Kronecker product. Now, we have the only term dependent on $\boldsymbol{\delta\theta}$ on the right hand side of the equation, which is necessary for matrix-vector calculus. This allows us to write

$$\frac{\partial \mathbf{x}_{Ci}}{\partial \boldsymbol{\delta\theta}} = -\left(\left(\mathbf{x}_{Bi}^T - \mathbf{t}_B^T\right)\left(\mathbf{T}_C^B\right)^- \otimes \mathbf{I}_{3\times 3}\right)\frac{\partial}{\partial \boldsymbol{\delta\theta}}\left\{\mathrm{vec}\left([\boldsymbol{\delta\theta}\times]\right)\right\} \tag{2.31}$$

which leaves only one term undefined. The last undefined term is the partial derivative of the skew-symmetric cross product matrix. Considering the form of the skew-symmetric cross product matrix, we can write this partial derivative as

$$\frac{\partial}{\partial \boldsymbol{\delta\theta}}\left\{\mathrm{vec}\left([\boldsymbol{\delta\theta}\times]\right)\right\} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.32}$$

which completely defines all of terms for the Jacobian for each measurement and allows us to update our state vector. In practice, since we have linearized our non-linear function, this is only an estimate of the optimal update. Therefore it is frequently necessary to iteratively refine the update by taking the newly estimated state and reprocessing the measurements through the steps described in this section.

3 Stereophotoclinometry for navigation

Stereophotoclinometry is a method by which a 3-Dimensional model of a planetary body can be made using only photos taken from a monocular camera. It was developed as a way to identify surface features on small planetary bodies, and created the added benefit that it provides a full 3D model that can be used for science analysis among other things. In the following sections we will examine in detail the processes that form the basis for SPC for navigation.

SPC for navigation can be easily broken down into fairly independent modules that each handle a specific part of the required processing. These modules and the overall architecture are shown in Fig. 8. As can be seen, the SPC for navigation modules primarily deal with the "image processing" section of the surface feature navigation steps shown in Fig. 1. Further, it should be apparent that the SPC processes require three primary inputs: the photo to be processed, the current best estimate of the spacecraft's state, and the model of the body being observed (in the form of maplets).

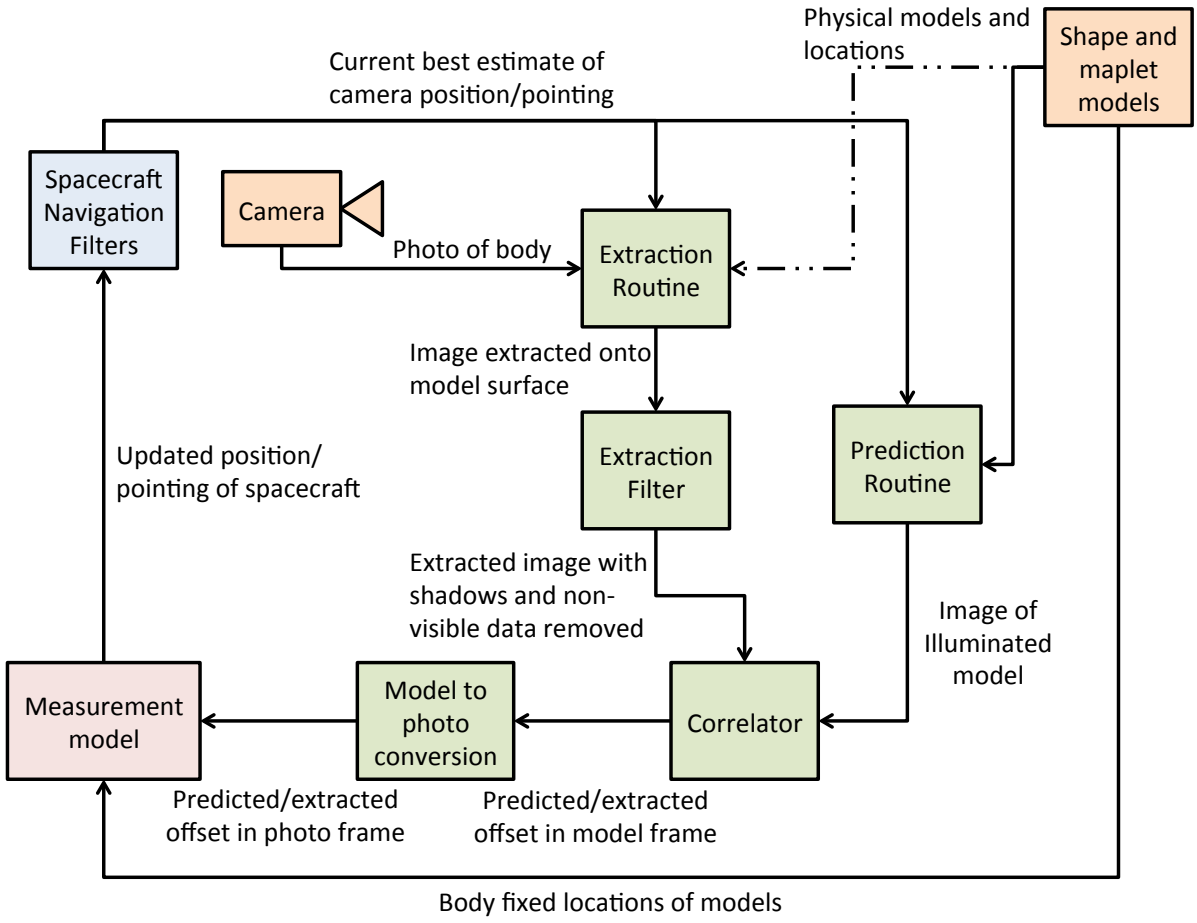


Figure 8: The SPC surface feature navigation architecture. Inputs to the process are shaded orange, the SPC modules are shaded green, the measurement model is shaded red, and the navigation filter is shaded blue.

3.1 SPC terminology

Before we continue, it is a good idea to formally define some of the terminology that will be used in the following sections (and was used sparingly in previous sections). In SPC there are two different physical models that are used to represent the surface of the planetary body. The first is a global shape model, which provides information about the full surface of the body being considered. It is generally comprised of vertex data all expressed in the asteroid-fixed frame, which provides the foundation for the model, and occasionally includes relative albedo data which can be used to illuminate the surface.

The other physical model is referred to as a maplet. A maplet is similar to the global shape model in that it provides the shape and relative albedo of the surface, but it is different in that it only covers a very small section of the surface. Since only a small section of the surface is considered, this makes it easier to have a much higher resolution in the maplets than can be achieved in the global shape model due to size constraints. Further, instead of representing the vertices in the asteroid-fixed frame as is done for the global shape models, the maplets express the vertices as elevation values above and below a local plane. This is where the maplet frame discussed in Sect. 1.1 comes into play. It is used to define the local coordinate system that the elevation data of the maplet is represented in. The $xy-$plane of the maplet frame is the local surface that the height data is defined in reference to. Furthermore, the origin of the reference frame expressed in the asteroid-fixed frame is defined as the landmark which is not necessarily inside of a distinguishable surface feature. In conjunction with the maplet frame is something that we will refer to as the maplet grid. The maplet grid is simply the m_x and m_y locations for which we have elevation data. It is referred to as a grid due to the fact that the m_x and m_y data only take on values of evenly spaced integers in the maplet frame. To see a demonstration of the maplet grid consult Fig. 9. Because of the layout of the grid, we will occasionally refer to the grid cells as maplet pixels. Finally, the maplets also contain a scaling term. The scaling term details how wide a maplet grid cell is in units of kilometers (or whatever unit of distance is being used).

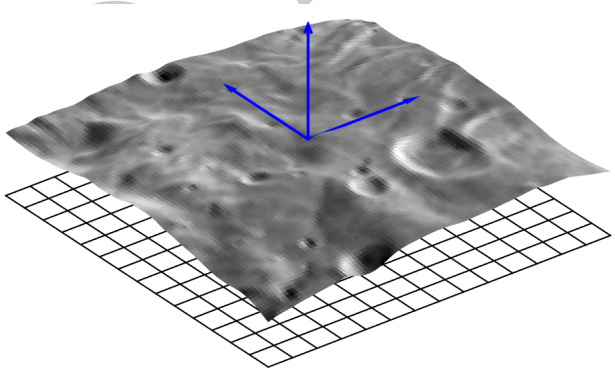


Figure 9: The maplet is made of height data, relative albedo data, a grid, a landmark, and a maplet frame. The maplet height data is represented by the surface shown here. The underlying maplet grid is shown beneath the height surface (the grid has been thinned out to show effect). The maplet frame is shown in blue, and the landmark is the origin of the maplet frame.

To further contrast these two different models we will now compare what each includes and does not include. The global shape model contains vertex data, connectivity information, and occasionally albedo data all expressed in the asteroid-fixed frame. The vertex data and connectivity information can be used to form a closed volume that represents the entire body being observed. The height variations in the global shape model are relatively low resolution. The maplet model contains height data along with implied connectivity information defined according to a local reference plane where the normal vector of the local reference plane is constrained to point in the same direction as the average of the surface normal information which can be computed from the maplet data. It always contains relative albedo data. In addition, information about the maplet frame is include in the maplet data. This includes the landmark (origin of the maplet frame expressed

in the asteroid-fixed frame) and the unit vectors pointing in the direction of the $x-$, $y-$, and $z-$ axes of the maplet frame expressed in the asteroid-fixed frame. The height data and connectivity information contained in the maplet cannot be used to form a closed volume, only a surface. It is generally much higher resolution then the global shape model. An example of the difference between the shape model and the maplets is shown in Fig. 10.

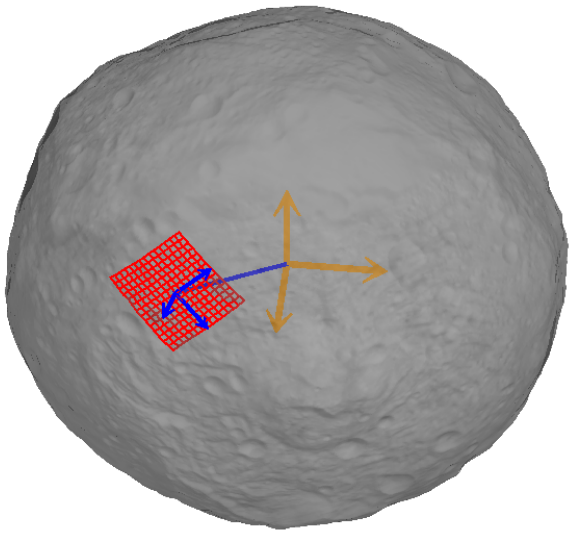


Figure 10: The maplet contains data about a small section of the surface, whereas the shape model contains the overall geometry of the body. In this Figure, the maplet is shown as a red grid, the shape model is the gray illuminated surface, the maplet frame are the blue axes and the asteroid-fixed frame are the orange axes.

The last term whose meaning may not be obvious is relative albedo. Albedo is a measurement of the reflectivity of a surface (i.e. how much of the incoming light is reflected versus absorbed by the surface). A relative albedo is a relative measurement of the reflectivity of a surface. A relative albedo map attempts to define how much more reflective sections of the surface are compared to others. We use relative albedos because it is difficult to calculate the true albedo of a surface with the only data being that obtained from an optical image of that surface, specifically because the SPC software does not contain any radiometry model. For instance, consider a mirror laying next to a rock. It is easy to say that the mirror obviously has an albedo that is higher than that of the rock, but without knowing the properties of the mirror and the rock or having an idea of the true response of the detector we cannot say what the actual albedo for either surface is.

3.2 Determining Which Maplets are Visible in a Photo

The first main step in SPC for navigation is to determine which maplets are visible in an photo (or at the very least, which maplets should be visible if we are where we think we are). This check depends almost entirely on the geometry of the scene according to the current best estimate of the spacecraft's position at the time the picture was taken. It requires three main inputs: maplet data, a specification of how much of the maplet must be visible, and the prior information about the spacecraft state at the time the photo was taken (also including the model for the camera that took the photo). Our first step is to use estimates to ignore landmarks that have no chance of being visible in the photo (whether because of the illumination conditions or the physical location of the landmark with respect to the camera). By using these simple and broad checks first, we decrease the amount of computational expense. After we have removed as many maplets as possible using these broad checks, we can then simply check to see if the predicted location of the maplet in the picture frame falls within the bounds of the photo. In addition, we can specify that only

a certain percentage of the maplet needs to be visible in the photo in order to process it. The steps used to decide which maplets are kept for processing follow.

The first step in picking maplets in the photo is to determine the geometry of the camera itself. This is done by calculating the half diagonal field of view (FOV) of the camera and by determining the line-of-sight direction of the vector from the center of the camera to the center of the camera pixel array in the asteroid-fixed frame. To calculate the half diagonal FOV consider the geometry in Fig. 11. As can be seen in the

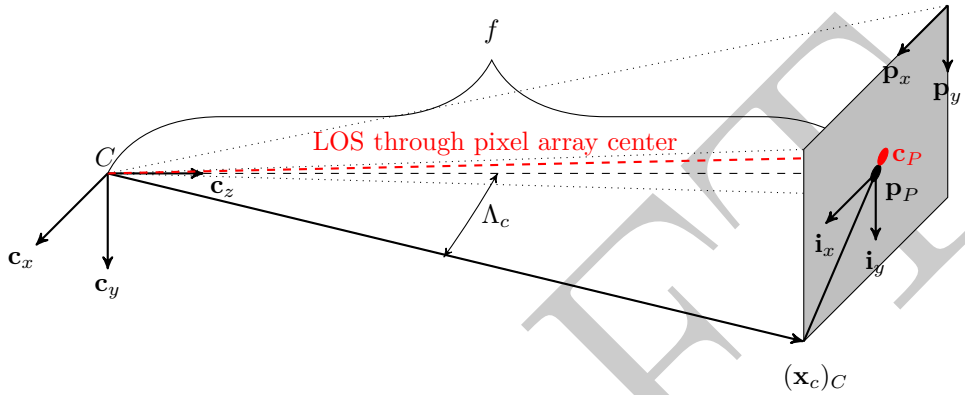


Figure 11: The geometry of the camera.

Figure, the half diagonal FOV is a measure of the angular distance of half the diagonal of the focal plane. In order to calculate the cosine of this angle we can use

$$\cos\Lambda_c = \frac{f}{\sqrt{(\mathbf{x}_c)_C^T(\mathbf{x}_c)_C}} \tag{3.1}$$

and to get the sine of the angle we can use the Pythagorean identity

$$\sin\Lambda_c = \sqrt{1-\cos^2\Lambda_c} \tag{3.2}$$

where Λ_c is the half diagonal FOV angle of the camera and $(\mathbf{x}_c)_C$ is the vector to the corner of the photo expressed in the camera frame. In addition to the half diagonal FOV, we also need the line-of-sight direction through the center of the pixel array expressed in the asteroid-fixed frame. To get the line-of-sight direction through the center of the pixel array, we take the unit vector in the direction of the center point expressed in the camera frame. Assuming our principal point is expressed in the picture frame, with the picture frame origin in the upper left hand corner of the photo (as shown in Fig. 3) then this is expressed as

$$\mathbf{c}_O = \mathbf{c}_P - \mathbf{p}_P \tag{3.3}$$

$$\mathbf{c}_C = \begin{bmatrix} K_x^{-1} & 0 & 0 \\ 0 & K_y^{-1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{c}_O \\ f \end{bmatrix} \tag{3.4}$$

$$\hat{\mathbf{b}}_B = \mathbf{T}_B^C \frac{\mathbf{p}_C}{\sqrt{\mathbf{p}_C^T\mathbf{p}_C}} \tag{3.5}$$

where $\mathbf{c}_P$ is the center point expressed in the picture frame, $\mathbf{c}_O$ is the center point expressed in the photo frame, $\mathbf{p}_P$ is the principal point expressed in the picture frame, $\mathbf{c}_C$ is the principal point expressed in the camera frame, and $\hat{\mathbf{b}}_B$ is the line-of-sight direction through the center of the camera expressed in the asteroid-fixed frame.

With the camera geometry defined we can now start considering the geometry of the rest of the scene. The first thing we need is the vector and distance from the spacecraft to the landmark expressed in the

asteroid-fixed frame. This is a simple addition:

$$(\mathbf{v}_{sc-lmk})_B = (\mathbf{v}_{sc-body})_B + (\mathbf{v}_{body-lmk})_B \tag{3.6}$$

$$d_{sc-lmk} = \sqrt{\mathbf{v}_{sc-lmk}^T \mathbf{v}_{sc-lmk}} \tag{3.7}$$

where $(\mathbf{v}_{sc-lmk})_B$ is the vector from the spacecraft to the landmark in the asteroid-fixed frame, $(\mathbf{v}_{sc-body})_B$ is the vector from the spacecraft to the body expressed in the asteroid-fixed frame, $(\mathbf{v}_{body-lmk})_B$ is the vector from the body to the landmark expressed in the asteroid-fixed frame, and d_{sc-lmk} is the distance between the spacecraft and the landmark expressed in units of distance. Now we can calculate the approximate maximum apparent angular size of the maplet. In order to do this we assume that the maplet frame $z-$axis is directly in line with the line-of-sight vector through the center of the pixel array and then calculate the angle between the apparent edge of the maplet in this orientation and the line-of-sight vector. The sine and cosine of this angle can be calculated as

$$\cos \Lambda_m = \frac{d_{sc-lmk}}{\sqrt{d_{sc-lmk}^2 + d_m/2}} \tag{3.8}$$

$$\sin \Lambda_m = \sqrt{1 - \cos^2 \Lambda_m} \tag{3.9}$$

where Λ_m is the angular half size of the maplet and d_m is the length of the sides of the maplet expressed in kilometers (or whatever distance unit is being used). With this data in hand we can now calculate the maximum angular distance that the landmark can be from the line-of-sight vector through the center of the pixel array of the camera for any of the maplet to possibly be in the field of view (note that even if the landmark center is within this distance it does not guarantee that any of the maplet will be visible in the photo). This is done by simply summing Λ_m and Λ_c. In practice, this is done through the use of the double angle formula as follows:

$$\cos(\Lambda_m + \Lambda_c) = \cos(\Lambda_c)\cos(\Lambda_m) - \sin(\Lambda_c)\sin(\Lambda_m) \tag{3.10}$$

Now we can begin checking for maplets that are not located within the photo.

The first check we perform is to check to see if the landmark is positioned such that any of the maplet could possibly be visible within the field of view. We do this by comparing the angle between the line-of-sight vector through the center of the pixel array and the line-of-sight vector to the landmark (calculated using a dot product) with the sum of the diagonal half FOV and the maplet half angle calculated in Eq. 3.10. That is we check if

$$\hat{\mathbf{b}}_B^T (\hat{\mathbf{v}}_{sc-lmk})_B > \cos(\Lambda_m + \Lambda_c) \tag{3.11}$$

where $(\hat{\mathbf{v}}_{sc-lmk})_B$ is the unit vector in the direction of $(\mathbf{v}_{sc-lmk})_B$. If the above is true then it means that the at least part of the maplet could be within the image. It is checking that the angle between the line-of-sight vector through the center of the pixel array and the landmark center is smaller than the maximum angular distance that can occur for any of the maplet to be visible (note that the fact that we are comparing cosines and not angles flips the direction of the greater than).

The next test we perform is to check the orientation of the maplet. We do this by taking the dot product of the maplet frame $z-$axis (roughly the normal vector of the entire maplet) with the line-of-sight vector from the spacecraft to the landmark (which gives us the cosine of the angle between the vectors). If this value is positive or smally negative then it means that the maplet is oriented in such a way that we are looking at it from behind it (positive) or nearly from along the $xy-$plane of the maplet frame. Mathematically we can perform this check as

$$-(\hat{\mathbf{v}}_{sc-lmk})_B^T (\hat{\mathbf{m}}_z)_B > 0.05 \tag{3.12}$$

where $(\hat{\mathbf{m}}_z)_B$ is the unit vector in the direction of the maplet frame $z-$axis expressed in the asteroid-fixed frame. If the above evaluates as true then it means that the maplet frame is positioned in such a way that it may be possible to view the surface from the camera. If it is not true then we can ignore this maplet for this image.

The next check is on the illumination of the maplet. Here we want to make sure that the sun is striking the top of the maplet, and not the side or bottom. In order to do this we once again check the dot product of the maplet frame $z-$axis, this time with the incoming sunlight vector. To check this mathematically we have

$$(\hat{\mathbf{m}}_z)_B^T \hat{\mathbf{s}}_B > 0 \tag{3.13}$$

where $\hat{\mathbf{s}}_B$ is the unit vector in the direction from the landmark to the sun expressed in the asteroid-fixed frame (this is input along with the state at the time the photo was taken). If the above is true then it means that the sun is above the maplet and it will be illuminated, and thus we can continue with the maplet. If it is not true then we should discard the maplet for this photo.

After checking the sunlight we can now check the distance resolution of the photo at the maplet (the amount of the surface that is covered by each pixel expressed in km/pix or some other unit of distance per pixel). For this check we want to be sure that the resolution of the image at the maplet is above some predefined threshold. The check for this case is easy

$$r_{img} > r_{thresh} \tag{3.14}$$

where r_{img} is the resolution of the photo at the maplet and r_{thresh} is the threshold resolution[18]. Calculating the resolution of the photo at the maplet is slightly more difficult. The first step is to calculate the derivative that relates the conversion from the maplet frame to the picture frame. This is done by using finite differencing. Therefore we have

$$\frac{\partial \mathbf{x}_P}{\partial m_x} = \frac{\mathbf{x}_P^+ - \mathbf{x}_P^-}{2\delta m_x} \tag{3.15}$$

where $\partial \mathbf{x}_P / \partial m_x$ is the change in the picture location with respect to a change along the maplet frame $x-$axis, $\mathbf{x}_P^+$ is the picture frame location of the landmark plus some step δm_x along the $x-$axis of the maplet frame, and $\mathbf{x}_P^-$ is the picture frame location of the landmark minus some step δm_x along the maplet frame $x-$axis. Here, $\mathbf{x}_P^\pm$ are found by taking the perturbed points in the asteroid-fixed frame and running them through the entire Owen camera model discussed in Section 1.1.1 to get the picture frame location of the points. This finite differencing is completed for each axis of the maplet frame and that gives the maplet to picture frame derivative. Once we have the maplet to picture frame derivative, we can continue with calculating the resolution of our photo at the maplet. We do this by calculating the area of a maplet grid cell in the picture plane, which is the determinant of the m_x and m_y terms of the maplet to picture derivative[19]. From here, we simply invert this value and multiply by the scaling term to go from units of maplet pixels to kilometers. That gives us the resolution of the photo at the landmark.

Our next check is to check the size of the photo and the maplet. To do this we simply want to be sure that the maplet is not larger than the area that we can see in our photo (because if it was it would be difficult to correlate with that maplet). We already know the length of a side of the maplet in units of distance (d_m) so all we need to get is the length of the sides of our photo at the surface. Determining the actual length of the sides at the surface is hard so instead we will make an estimate of their size (remember at this point we are still using broad strokes to remove maplets that we do not want to spend a lot of time checking to see if they are in the photo or not). To get the approximate length of the sides of our photo at the surface, we can return to the photo resolution at the landmark. Since the photo resolution at the landmark approximately tells us the size of one pixel at the landmark, if we multiply this value by the length of a side in pixels (or by the length of a side if the area of the photo was formed by a square) then we have an approximate of the size of our photo of the surface. Once we have this our check is simply

$$d_m < d_p \tag{3.16}$$

where d_p is the approximate length of a side of the photo at the landmark (by this we mean the distance

[18]Note that we only check the resolution of the photo at the landmark if we have already found a lot of other maplets that are within the photo and we are trying to limit the amount of memory we are using. By default the resolution is checked starting at 200 maplets; however, this value can be changed by setting the NUMLM value in the INIT_LITHOS file.

[19]When we have a Jacobian matrix that approximates a transformation from one coordinate system to another, part of the Jacobian will rotate into the new coordinate frame and part will scale the results. Since we know that all rotation matrices have to have a unit determinant, then we know that the scaling portion of the Jacobian is the value of its determinant, and that is why we can claim that the determinant of the maplet to picture frame Jacobian is the size of a maplet grid cell in the picture frame.

along the surface that is visible in the image). This completes the broad check that we are using to throw out landmarks that have no chance of being seen in our image.

We can now attempt to actually determine whether a landmark is visible within the photo or not. This will require at most 9 checks. First we will check to see if the landmark location falls within the bounds of the photo. To do this we simply run the landmark location in the asteroid-fixed frame through the Owen camera model described in Section 1.1.1 and see if the resulting picture location is within the bounds of our photo. If the landmark location is within the bounds of our photo then this landmark is visible in our photo and we continue to the next step with it. If it is not within the bounds of our photo then we can check to see if a fraction of the landmark is within the photo. In order to do this, we simply set up a box of points at some specified fractional distance along the maplet frame $x-$ and $y-$axes and run them through the Owen camera model. If any of these fall within our photo bounds then we keep the maplet and move onto the next step. If none of these are within the photo then the maplet is thrown away. Mathematically the points we will run through the Owen camera model are

$$\mathbf{p}_B = (\mathbf{v}_{body-lmk})_B d_m w(k\hat{\mathbf{m}}_x + j\hat{\mathbf{m}}_y),\ j,k = -1,0,1 \tag{3.17}$$

where w is the fractional width which is specified by the user and $\mathbf{p}_B$ are the points that we will check to see if they lie in the photo. Note here that this is actually something of an inverse fractional width. That is, if we want 90% of half of the maplet to be visible in the photo then we need to set $w = 0.1$ not $w = 0.9$.

3.2.1 Checking For Hidden Maplets

We have now identified maplets where at least part of the maplet is within the field of view. We have not checked, however, to be sure that the portion of the maplet within the field of view is actually visible. Therefore we now need to check whether the maplet is occluded by some other part of the shape or not. In order to check whether our maplet is occluded by other parts of the surface we have a few different steps. First we identify the picture frame position of our landmark using the Owen camera model. We now work backwards through the Owen camera model to get a line-of-sight vector in the direction of the photo frame position of the landmark. In the case where we have a distortion model for our camera (which is always the case for real life images) this is a complicated process that requires an iterative solution. Once the line-of-sight vector has been determined, the next step is to determine points on the surface of the body that fall along the line-of-sight vector. Finally, we compare the closest point to the spacecraft that falls along the line-of-sight vector to the location of our landmark. If these points are relatively close (within one percent of the maximum diameter of the body) then the landmark is the first object along the line-of-sight vector and it is not obscured by the rest of the surface. If the points are far away from each other then the landmark is occluded by some other portion of the body and thus should not be used. We now discuss these steps in detail.

The first step of getting the picture frame location of the maplet was already discussed in great detail in Sect. 1.1.1. Returning from the picture frame location to a line-of-sight vector still needs discussion. In order to do this, we must go from the distorted picture frame location of the landmark to the undistorted photo frame location of the landmark (note that the undistorted photo frame location has also been converted from units of pixels to units of distance), which entails a frame transformation, a unit transformation, and the removal of a non-linear distortion model. There is not an easy analytic solution to do this so instead we can use an iterative solution. First we set an initial guess of the undistorted photo frame location of the landmark to be at the distorted picture frame location of the landmark. We do this by moving from the picture frame to the photo frame and converting from units of pixels to units of distance. That is

$$\mathbf{x}_I^{(1)} = \begin{bmatrix} K_x^{-1} & 0 \\ 0 & K_y^{-1} \end{bmatrix} (\mathbf{x}_P - \mathbf{p}_P) \tag{3.18}$$

where $\mathbf{x}_I^{(1)}$ is the initial guess of our undistorted photo frame location and $\mathbf{x}_P$ is the true distorted picture frame location of our point. From here we can now apply the distortion model and the conversion to units of pixels by going through Eqs. 1.15 and 1.16. Once we have transformed our initial undistorted photo frame

location guess into a distorted location in the picture frame we can start our iterative updates. This is done using

$$\mathbf{x}_I^{(k)} = \mathbf{x}_I^{(k-1)} + \begin{bmatrix} K_x^{-1} & 0 \\ 0 & K_y^{-1} \end{bmatrix} \left(\mathbf{x}_P - \mathbf{x}_P^{(k-1)} \right) \tag{3.19}$$

where $\mathbf{x}_I^{(k)}$ is the k^{th} estimate of the undistorted photo frame location of our point, $\mathbf{x}_I^{(k-1)}$ is the previous estimate of the undistorted photo frame location of our point, and $\mathbf{x}_P^{(k-1)}$ is the distorted picture location of our previous estimate. We now return our undistorted photo frame location guess into a distorted picture frame location guess by using Eqs. 1.15 and 1.16 and repeat the process. We iterate this process until the difference between our true distorted picture frame location and the distorted picture frame guess is less than some tolerance (1×10^{-8} in the SPC source code).

Once we have our undistorted photo frame location we can now calculate the line-of-sight vector to the landmark. We do this by taking the unit vector in the direction of the vector pointing from the center of the camera frame to the undistorted photo frame location. That is

$$(\hat{\mathbf{v}}_{sc-lmk})_C = \frac{\begin{bmatrix} x_I^{(k)} & y_I^{(k)} & f \end{bmatrix}^T}{\left\| \begin{bmatrix} x_I^{(k)} & y_I^{(k)} & f \end{bmatrix}^T \right\|} \tag{3.20}$$

where $(\hat{\mathbf{v}}_{sc-lmk})_C$ is the line-of-sight vector from the spacecraft to the landmark, $x_O^{(k)}$ and $y_O^{(k)}$ are the $x-$ and $y-$components of the final undistorted photo frame location estimate of the point, and $\| \bullet \|$ indicates the $2-$norm of a vector.[20] We can now take our line-of-sight vector and determine the point that it strikes on the surface of the body.

To determine the point that we strike first on the body along the line-of-sight vector coming out of our camera we have to turn to the technique of ray tracing. Ray tracing is a popular technique for many different optical applications and computer graphics problems. We will now describe the algorithm that is used in the SPC source code. The first thing we want to do is to define some specialized frame that uses the line-of-sight vector as the $z-$axis for reasons that will become apparent later. Therefore let us define the following frame, which we will henceforth refer to as the tilde frame:

$$\widetilde{\mathbf{z}}_C = (\hat{\mathbf{v}}_{sc-lmk})_C \qquad \widetilde{\mathbf{y}}_C = \frac{\widetilde{\mathbf{z}}_C \times (\mathbf{b}_x)_C}{\|\widetilde{\mathbf{z}}_C \times (\mathbf{b}_x)_C\|} \qquad \widetilde{\mathbf{x}}_C = \frac{\widetilde{\mathbf{y}}_C \times \widetilde{\mathbf{z}}_C}{\|\widetilde{\mathbf{y}}_C \times \widetilde{\mathbf{z}}_C\|} \tag{3.21}$$

where $\widetilde{\mathbf{x}}_C$, $\widetilde{\mathbf{y}}_C$, and $\widetilde{\mathbf{z}}_C$ are the unit vectors in the direction of the axes of the tilde frame expressed in the camera frame, $\bullet \times \bullet$ indicates a cross product between two vectors, and $(\mathbf{b}_x)_C$ is the unit vector in the direction of the asteroid-fixed frame $x-$axis expressed in the camera frame.[21] This frame is shown in Fig. 12.

With the tilde frame defined we now wish to express all of the vertices of our shape model in this frame. We do this by getting the vectors from the spacecraft to the vertices, rotating these vectors into the camera frame, and then rotating them into the tilde frame. That is

$$\widetilde{\mathbf{v}}_{sc-vert}^{(i)} = \mathbf{T}_{\sim}^C \mathbf{T}_C^B \left[(\mathbf{v}_{sc-body})_B + \left(\mathbf{v}_{body-vert}^{(i)} \right)_B \right] \tag{3.22}$$

where $\widetilde{\mathbf{v}}_{sc-vert}^{(i)}$ is the vector from the spacecraft to the i^{th} vertex expressed in the tilde frame, $\mathbf{T}_{\sim}^C$ is the rotation matrix from the camera frame to the tilde frame given as

$$\mathbf{T}_{\sim}^C = \begin{bmatrix} \widetilde{\mathbf{x}}_C & \widetilde{\mathbf{y}}_C & \widetilde{\mathbf{z}}_C \end{bmatrix}^T, \tag{3.23}$$

[20] It is unclear why we must go through the act of projecting the landmark location onto the photo plane, and then un-distorting this projected location to get our line-of-sight when we could have just defined the line-of-sight vector as $(\hat{\mathbf{v}}_{sc-lmk})_C = \mathbf{T}_C^B \left[\left(\mathbf{v}_{sc-body} \right)_B + \left(\mathbf{v}_{body-lmk} \right)_B \right] / \left\| \left[\left(\mathbf{v}_{sc-body} \right)_B + \left(\mathbf{v}_{body-lmk} \right)_B \right] \right\|$ using information that we already know. Furthermore this line-of-sight vector is likely more accurate since we have not had to use the iterative solution to get to the un-distorted photo frame location of the point. Anyway, I digress...

[21] It does not really matter how we define the $\widetilde{\mathbf{y}}_C$ or $\widetilde{\mathbf{x}}_C$ axes so long as they form a right-handed coordinate system and are all orthogonal to each other.

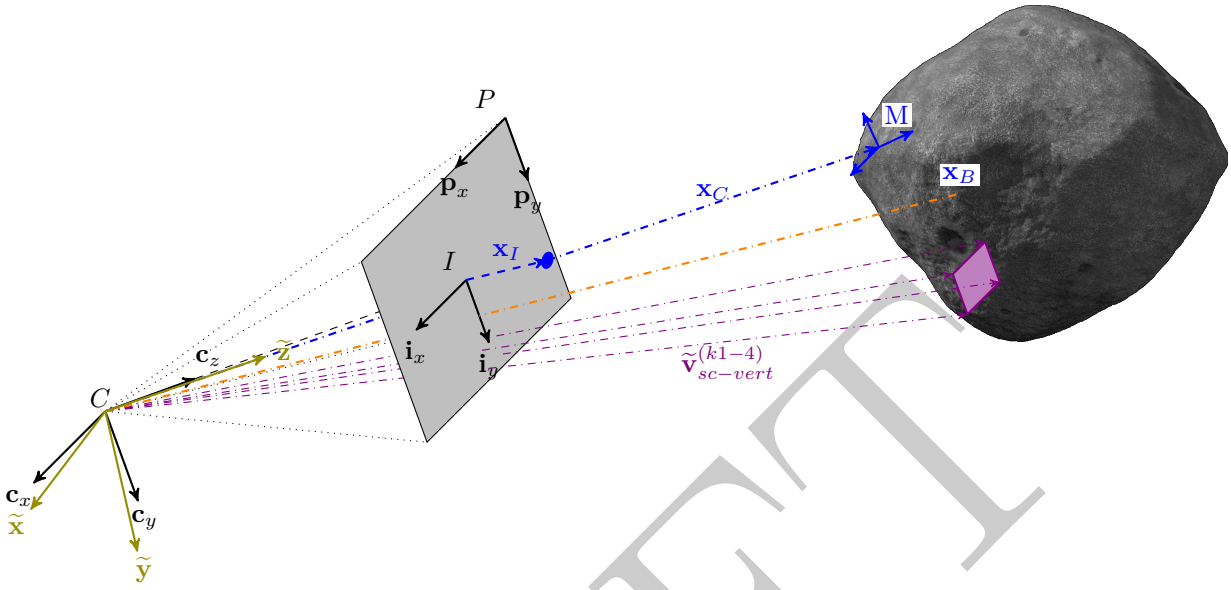


Figure 12: The geometry that is involved in the ray tracing. In this figure, all vectors having to do with the landmark location are shown in blue, the tilde frame is shown in olive, an example grid from the shape model is shown in violet, the vectors involving the center of the asteroid's body are drawn in orange, and the distance we are comparing is shown in green. Note that this distance is the orthogonal distance from the grid vertices to the $z-$axis of the tilde frame.

and $\left(\mathbf{v}^{(i)}_{body-vert}\right)_B$ is the asteroid-fixed vector to the i^{th} vertex.

After defining all of our spacecraft to vertex points in the tilde frame it is now possible to begin determining where we strike the surface along our line-of-sight vector. Our first step is to throw away grids that have no chance of falling along the line-of-sight vector (this helps to decrease the amount of computations we need to complete later by throwing out a lot of the data with a relatively efficient step). This involves calculating the maximum diagonal distance of all of the grid points, that is finding

$$d^2_{max} = \max_{k,ij} \left\{ \left[\left(\mathbf{v}^{(ki)}_{body-vert}\right)_B - \left(\mathbf{v}^{(kj)}_{body-vert}\right)_B \right]^T \left[\left(\mathbf{v}^{(ki)}_{body-vert}\right)_B - \left(\mathbf{v}^{(kj)}_{body-vert}\right)_B \right] \right\} \tag{3.24}$$

where $\left(\mathbf{v}^{(ki)}_{body-vert}\right)_B$ is vertex i of grid k expressed in the body frame, $\left(\mathbf{v}^{(kj)}_{body-vert}\right)_B$ is vertex j of grid k expressed in the body frame, $\max_{k,ij}\{\bullet\}$ indicates to take the maximum value varying k and ij pairs, and d^2_{max} is the square of the maximum distance covered by any grid cell. Note that because of the way the grid cell vertices are labeled (see Fig. 13), we only ever need to check two different ij pairs for each cell, $(i = 1, j = 3)$ and $(i = 2, j = 4)$.[22] Now, with our maximum cell distance in hand, we can throw out any cells that have at least one point with an orthogonal distance to the tilde frame $z-$axis greater than the maximum cell distance. That is we want to remove any cell k where

$$\max_{1\le i\le 4} \left\{ \begin{bmatrix} \widetilde{x}^{(k)}_i & \widetilde{y}^{(k)}_i \end{bmatrix} \begin{bmatrix} \widetilde{x}^{(k)}_i \\ \widetilde{y}^{(k)}_i \end{bmatrix} - d^2_{max} \right\} > 0 \tag{3.25}$$

is true. Here $\widetilde{x}^{(k)}_i$ and $\widetilde{y}^{(k)}_i$ are the $x-$ and $y-$components of the i^{th} vertex of cell k expressed in the tilde frame (see Fig. 13).

[22] We could perform this operation in whatever frame we want since all of our steps are linear transformations. I have presented it here in the body frame because that is how it is done in the SPC source code.

Having removed many of our cells we can now determine which cells are actually pierced by our line-of-sight vector. To do this consider Fig. 13. In this Figure we see a cell with different possible intersection points for our line-of-sight vector (we have projected everything onto the $xy-$plane of the tilde frame). Observe the angles between the vectors from the interception of the line-of-sight vector (tilde frame $z-$axis) to the corners. In the first cell all of the angles proceed in the same angular direction (counter clockwise). In the remaining cells there is always one angle that proceeds in the opposite direction to the others (clockwise). This implies that we can check for intersection with a cell by considering the cross products of the vectors from the intersection of the line-of-sight vector with the cell plane to the corners of the cell. If all of these cross products point in the same direction (particularly if they all point in the opposite direction as the line-of-sight vector[23]) then the line-of-sight vector pierces the cell, otherwise it does not. Therefore, all we have to do is consider the cross product of the 2D vectors from the tilde frame $z-$axis to the cell corners projected onto the $xy-$plane of the tilde frame (done by setting the $z-$component of the $\widetilde{\mathbf{v}}^{(ki)}_{sc-vert}$ vectors equal to 0) and see if the $z-$components (which will be the only components) are all negative. That is, we want to keep cells with

$$\min_{1\leq i\leq 4}\left\{\widetilde{x}_i^{(k)}\widetilde{y}_{i+1}^{(k)}-\widetilde{x}_{i+1}^{(k)}\widetilde{y}_i^{(k)}\leq 0\right\}=1 \tag{3.26}$$

where $\widetilde{x}_i^{(k)}$ and $\widetilde{y}_i^{(k)}$ refer to the $x-$ and $y-$components of the i^{th} corner of the k^{th} cell expressed in the tilde frame, the grid cell points are defined as shown in Fig. 13, and when i is 4 then $i+1$ is 1.

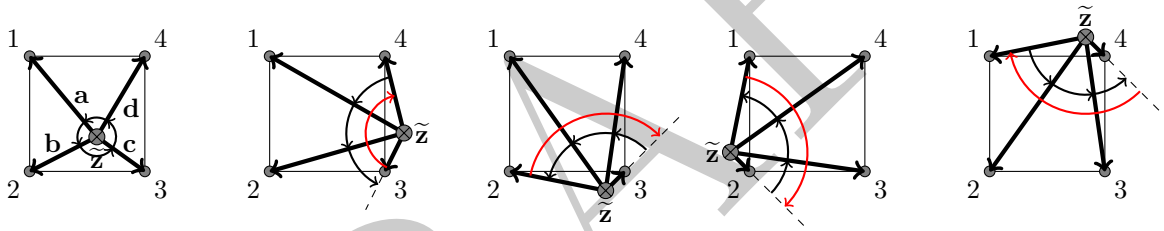


Figure 13: Projections of a shape model cell projected onto the $xy-$plane of the tilde frame with possible locations of the tilde frame $z-$axis. We can use a cross product to identify cells where all of the angles are pointing in the opposite direction of the tilde frame $z-$axis. In these diagrams the tilde frame $z-$axis is going into the paper.

With the cells that are pierced identified all we need to do is determine how far along the line-of-sight vector these intersection points are. To do this we need to interpolate the $z-$component values when the $x-$ and $y-$components are equal to 0 (since the $z-$axis of the tilde frame points along the line-of-sight vector). In the SPC code bilinear interpolation is used to get the $z-$components which also represent the distance between the center of the camera and the intersection point.

Bilinear interpolation is a 3D interpolation technique that attempts to fit the product of two linear functions to data points in a cell (therefore, despite the name, a bilinear interpolation is not a linear function). It is a relatively simple and efficient interpolation technique and provides plenty of accuracy for our needs in this case. There is one problem though. Bilinear interpolation only works when the data is presented in a true grid format. That is, it only works when the data points we are trying to fit to are rectangular or square in the $xy-$coordinates of our fit. To visualize this examine Fig. 14.

Since our data is not regularly gridded, we cannot use a standard bilinear interpolation. Instead we have to use 3 different bilinear interpolations, one on the $x-$data, one on the $y-$data, and one on the $z-$data. To begin, let us assume that the four corners of the cell that we are considering are now expressed in a 5D space where we have axes of $\boldsymbol{\iota}$, $\boldsymbol{\gamma}$, $\widetilde{\mathbf{x}}$, $\widetilde{\mathbf{y}}$, and $\widetilde{\mathbf{z}}$. Now let us assume that each $\boldsymbol{\iota\gamma}$ pair corresponds to some

[23]The vectors will only point in the opposite direction of the line-of-sight vector if the cell corners are defined in a counter clockwise manner when examined along the line-of-sight vector. This will always be the case for cells whose normal vectors are pointing in the opposite direction of the line-of-sight vector due to the definition of a cell in the ICQ format. These are the only cells we would want to consider anyway so this works to our advantage.

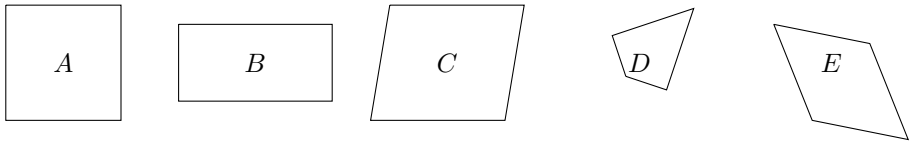


Figure 14: Bilinear interpolation only works when the x and y data is gridded as is the case in blocks A and B (assuming we are trying to fit the z data). If we want to fit scattered data (blocks $C-E$) we need to use other tricks to try and make the data gridded.

point in our 3D tilde frame. Further, let us define that the following $\boldsymbol{\iota\gamma}$-$\widetilde{\mathbf{x}}\widetilde{\mathbf{y}}\widetilde{\mathbf{z}}$ pairings:

$$(0,0) \rightarrow \begin{bmatrix} \widetilde{x}_1^{(k)} \\ \widetilde{y}_1^{(k)} \\ \widetilde{z}_1^{(k)} \end{bmatrix} \qquad (1,0) \rightarrow \begin{bmatrix} \widetilde{x}_4^{(k)} \\ \widetilde{y}_4^{(k)} \\ \widetilde{z}_4^{(k)} \end{bmatrix}$$

$$(0,1) \rightarrow \begin{bmatrix} \widetilde{x}_2^{(k)} \\ \widetilde{y}_2^{(k)} \\ \widetilde{z}_2^{(k)} \end{bmatrix} \qquad (1,1) \rightarrow \begin{bmatrix} \widetilde{x}_3^{(k)} \\ \widetilde{y}_3^{(k)} \\ \widetilde{z}_3^{(k)} \end{bmatrix}$$

where $(\boldsymbol{\iota}, \boldsymbol{\gamma})$ are the $\boldsymbol{\iota\gamma}$ pairs and $\begin{bmatrix} \widetilde{x}_i^{(k)} & \widetilde{y}_i^{(k)} & \widetilde{z}_i^{(k)} \end{bmatrix}^T$ are the corner points of our shape cells expressed in the tilde frame. We are now working with gridded data in the $\boldsymbol{\iota\gamma}$ plane therefore we can perform bilinear interpolation in the $\boldsymbol{\iota\gamma}$ plane on the components of our corner points expressed in the tilde frame. Now, since we know we are looking for the point that corresponds to $\widetilde{x} = 0$ and $\widetilde{y} = 0$ we can set up a system of equations to solve for the $\boldsymbol{\iota\gamma}$ pair that corresponds to this point, and then use the calculated $\boldsymbol{\iota\gamma}$ pair to calculate the interpolated $\widetilde{z}$ value at this point (see Fig. 15 for a visual layout of what is happening). This leaves us the following system of two equations with two unknowns

$$0 = X_0 + X_1\iota_0 + X_2\gamma_0 + X_3\iota_0\gamma_0 \tag{3.27}$$

$$0 = Y_0 + Y_1\iota_0 + Y_2\gamma_0 + Y_3\iota_0\gamma_0 \tag{3.28}$$

where

$$X_0 = \widetilde{x}_1^{(k)} \qquad X_1 = \widetilde{x}_4^{(k)} - \widetilde{x}_1^{(k)} \qquad X_2 = \widetilde{x}_2^{(k)} - \widetilde{x}_1^{(k)} \qquad X_3 = \widetilde{x}_1^{(k)} - \widetilde{x}_2^{(k)} - \widetilde{x}_4^{(k)} + \widetilde{x}_3^{(k)} \tag{3.29}$$

$$Y_0 = \widetilde{y}_1^{(k)} \qquad Y_1 = \widetilde{y}_4^{(k)} - \widetilde{y}_1^{(k)} \qquad Y_2 = \widetilde{y}_2^{(k)} - \widetilde{y}_1^{(k)} \qquad Y_3 = \widetilde{y}_1^{(k)} - \widetilde{y}_2^{(k)} - \widetilde{y}_4^{(k)} + \widetilde{y}_3^{(k)} \tag{3.30}$$

are the standard bilinear interpolation coefficients for the $\widetilde{x}$ and $\widetilde{y}$ values. It is easy to solve this system of equations for ι_0 and γ_0. The results are as follows:

$$\iota_0 = \begin{cases} -\frac{C}{B} - \frac{AC^2}{B^3}, & A \leq 1 \times 10^{-8} \\ \frac{-B+\sqrt{B^2-4AC}}{2A}, & A > 1 \times 10^{-8} \end{cases} \tag{3.31}$$

$$\gamma_0 = \begin{cases} -\frac{X_0+X_1\iota_0}{X_2+X_3\iota_0}, & |X_2 - X_3\iota_0| > |Y_2 + Y_3\iota_0| \\ -\frac{Y_0+Y_1\iota_0}{Y_2+Y_3\iota_0}, & |X_2 - X_3\iota_0| \leq |Y_2 + Y_3\iota_0| \end{cases} \tag{3.32}$$

where

$$A = X_1Y_3 - X_3Y_1 \qquad B = X_1Y_2 - X_2Y_1 - X_3Y_0 + X_0Y_3 \qquad C = X_0Y_2 - X_2Y_0 \tag{3.33}$$

are the coefficients of the quadratic function of ι_0 found by solving the system of equations given in Eqs. 3.27 and 3.28[24].

[24] It is not apparent where the first equation for ι_0 comes from. If we knew A had to be positive (and thus the logic is checking if A is small) then I can see the first term there $(-C/B)$ which is just the solution of the linear equation assuming A is 0. I have not seen the second term before though and I cannot figure out how to get to it. Will have to ask Dr. Gaskell about this. Similarly I am not sure about the two different equations for γ_0. I know where they come from (solving either Eq. 3.27 or Eq. 3.28 for γ_0) but I am not sure why using one or the other would be beneficial. I expect that this is related to the logic on A as well.

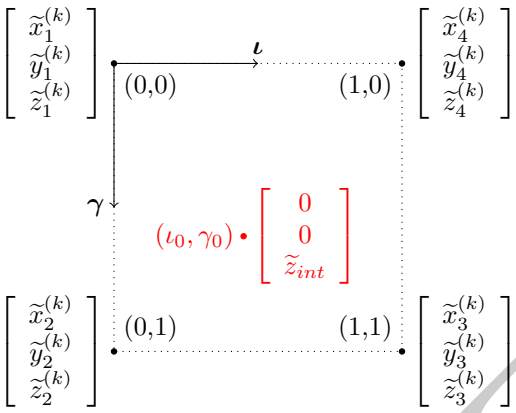


Figure 15: In order to use bilinear interpolation we need to work in a temporary imaginary 5D space. It is easier to think of this as something like a 3D vector field projected onto a plane. In this image we are looking at the $\iota\gamma$ plane where each point corresponds to a surface point expressed in the tilde frame. The point we are interested in is when $\widetilde{x}$ and $\widetilde{y}$ are equal to 0. This point is shown in red.

With ι_0 and γ_0 in hand it is now relatively straight forward to calculate the distance between the camera and the intercept point of the cell by the line-of-sight vector using a standard bilinear interpolation on the $\widetilde{z}_i$ data. As long as ι_0 and γ_0 are between 0 and 1 (that is as long as they fall inside of our imaginary $\boldsymbol{\iota\gamma}$ grid) then our intersection distance is given by

$$\widetilde{z}_{int} = Z_0 + Z_1\iota_0 + Z_2\gamma_0 + Z_3\iota_0\gamma_0 \tag{3.34}$$

where

$$Z_0 = \widetilde{z}_1^{(k)} \qquad Z_1 = \widetilde{z}_4^{(k)} - \widetilde{z}_1^{(k)} \qquad Z_2 = \widetilde{z}_2^{(k)} - \widetilde{z}_1^{(k)} \qquad Z_3 = \widetilde{z}_1^{(k)} - \widetilde{z}_2^{(k)} - \widetilde{z}_4^{(k)} + \widetilde{z}_3^{(k)} \tag{3.35}$$

are the standard bilinear interpolation coefficients for the $z-$data.

We perform these steps for every cell which is intersected by the line-of-sight vector. Once we have all of our distances then the minimum distance is the first intersection. If we compare this intersection point with the landmark location then we can check to see whether the center of the landmark is occlude by other surface geometry or not. This concludes the invisible checking of the landmarks.

3.3 Preparing the "Images"

Having selected which maplets are visible in our photo, we can now turn to examining the mathematical foundation of SPC for navigation. The first grouping of routines are focused on preparing "images" that will be used to calculate the required offsets between our predicted locations and actual locations. We quote images here because we are actually not working with images in the original sense of the word. Instead we are working with surfaces that also contain intensity values. The surfaces we are working with are not flat, and in fact are the maplets themselves. Therefore, for each point in the grid of one of our "images" we have both a height value and an illumination value. In practice, these images are always shown from above and thus they look like flat images that we are used to seeing[25]. However, remembering that these are not flat images will be key to explain some of the steps that may seem out of place in the following discussion[26]. In order to help make this point clear consider Fig. 16 which shows both the flat image that we are usually presented with and the flat image projected onto the maplet shape that represents what we are actually working with.

[25]From this point forward I have dropped the quotes around image when referring to these illuminated surfaces, mostly because my pinky finger gets tired enough as is when writing latex without having to add quotes around a word numerous times

[26]I believe that it may actually be better to show the images at an angle to drive home the fact that they are not flat, but that's just me

In addition, this is why we have used photo in the rest of the paper in place of image. Throughout we will refer to the entire scene captured by the camera as a photo, photograph, or picture, and the maplet surfaces with illumination data as images.

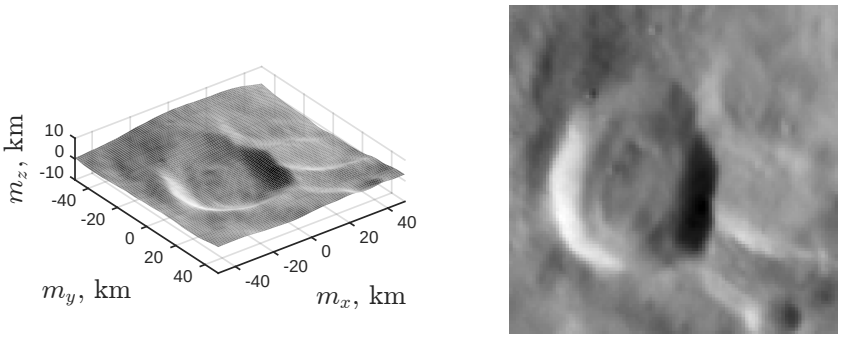


Figure 16: When we talk about the predicted and extracted images of a maplet, we are actually referring to the illumination values projected onto the maplet surface (as seen in the left plot) despite the fact that we frequently show the illumination data on a flat surface (the right plot).

3.3.1 Image prediction

To begin our discussion of preparing the images we will start by considering the steps used to predict our image. We start here in hopes that it will help to make the non-flat surface image idea more clear. In our prediction step, all we are doing is illuminating a maplet using a given illumination model at each grid in the maplet frame. Since we do not necessarily know much about the composition of the surface this essentially boils down to a geometry problem which is shown in Fig. 17. The steps simply entail (1) calculating the localized normal vectors at each point, (2) calculating the incidence angles for each maplet grid point (angle between the local normal vectors and the incoming direction of the collimated light source), (3) calculating the reflection angle (the angle between the direction vector to the camera and the local normal vector), (4) calculating the phase angle (angle between the incidence vector and the reflection vector), (5) getting the illumination using the illumination model, and (6) adjusting the average brightness to match the extracted image.

The maplet data that we have does not include local normal vectors for each grid location, so we need some way to calculate them. This is done by considering the gradients of the height data in the $x-$ and $y-$directions of the maplet frame. Obtaining the gradient of the height data is a straight forward application of finite differencing. We first do finite differencing on the height data in the x direction (column direction) and then do finite differencing in the vertical direction (row direction)[27]. We can then use the gradients at each grid point to determine the vectors in the direction of the surface in the $x-$ and $y-$maplet frame directions. To visualize this consider the fact that at maplet grid location k, to the best of our knowledge a step of Δx in the maplet grid $x-$direction will yield a change of $\Delta x \partial h_k/\partial x_k$, thus a vector in the $x-$direction of the surface at location k is given by

$$\mathbf{v}_x = \begin{bmatrix} 1 & 0 & \partial h_k/\partial x_k \end{bmatrix}^T, \tag{3.36}$$

where $\mathbf{v}_x$ is the vector in the direction of the surface along the $x-$direction of the maplet frame, $\partial h_k/\partial x_k$ is the gradient of the height data in the $x-$direction of the maplet frame, and we have let $\Delta x = 1$. This works similarly for the $y-$direction of the maplet frame. Since these two vectors lie along the surface to the best of our knowledge, the cross product will give us the normal vector to the surface at that point. Because of

[27] In the SPC source code there are a few different ways the finite differences are done. We will not discuss these here as they are all just different variations of finite differencing that use different chunks of the height data depending on what is available.

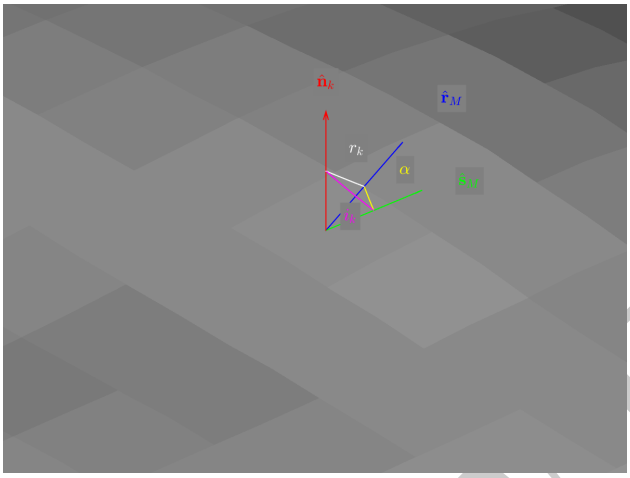


Figure 17: The illumination model used in SPC is primarily a function of the geometry. Here, $\hat{\mathbf{r}}_M$ is the unit vector pointing to the camera, $\hat{\mathbf{s}}_M$ is the unit vector pointing to the sun, and $\hat{\mathbf{n}}_k$ is the local normal vector. Angle i_k is the incidence angle and r_k is the reflectance angle. Note that all the vectors are pointing away from the surface here.

the unique structure of this problem the cross product of these vectors simply becomes

$$\mathbf{n}_k = \begin{bmatrix} -\frac{\partial h_k}{\partial x_k} & -\frac{\partial h_k}{\partial y_k} & 1 \end{bmatrix}^T \tag{3.37}$$

where $\mathbf{n}_k$ is the normal vector for grid location k. To visualize this, see Fig. 18.

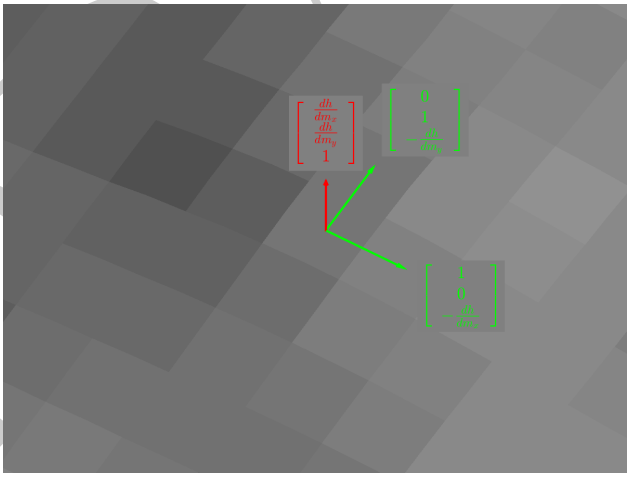


Figure 18: The normal at each surface point can be found by taking the cross product of the x and y gradient vectors at that point.

Having calculated the local normal vectors at each grid location for the maplet, the angles for steps 2-4 can be easily found using the dot product of the vectors involved. Therefore we have

$$\cos i_k = \hat{\mathbf{n}}_k^T \hat{\mathbf{s}}_M \tag{3.38}$$

$$\cos r_k = \hat{\mathbf{n}}_k^T \hat{\mathbf{r}}_M \tag{3.39}$$

$$\alpha = \cos^{-1}(\hat{\mathbf{s}}_M^T \hat{\mathbf{r}}_M) \tag{3.40}$$

where i_k is the incidence angle at location k, r_k is the reflectance angle at location k, α is the phase angle, $\hat{\mathbf{n}}_k$ is the local unit normal vector (unit vector in the direction of the local normal vector) at location k, $\hat{\mathbf{s}}_M$

is the unit vector pointing in the direction from the body to the sun expressed in the maplet frame, and $\hat{\mathbf{r}}_M$ is the unit vector pointing in the direction from the landmark to the camera expressed in the maplet frame.

Once the geometry has been calculated, all that remains is to illuminate the maplet using the illumination model and then to adjust the average brightness of the predicted image to match that of the extracted image. This is done using one of two illumination models which are presented here in brevity [28]. The first illumination model that can be used (and the one that is used the most) is the McEwen model [4]. The McEwen model is given by

$$I_k = a_k \left((1-\beta)\cos i_k + \beta \frac{\cos i_k}{\cos i_k + \cos r_k} \right) \tag{3.41}$$

where I_k is the illumination value for location k, a_k is the albedo value for location k, and $\beta = \exp(-\alpha/\alpha_0)$ is a weighting term between Lambertian and Lommel-Seeliger scattering terms with $\alpha_0 = 60$[29]. The weighting term was chosen to approximate McEwen's lunar function [5]. The second illumination model is exactly the same as the McEwen model, however it sets the value of $\beta = 0.65$, ignoring the dependence on the phase angle.

The final step is to adjust the average brightness of the predicted image to match that of the extracted image. This is (in its simplest form) done by multiplying the predicted image by the ratio between the average illumination of the extracted image and the average illumination of the predicted image calculated using Eq. 3.41:

$$\lambda = \frac{\mu_e}{\mu_p} \tag{3.42}$$

where lambda is the scaling term, μ_e is the average illumination of the extracted image, and μ_p is the average illumination of the image predicted by Eq. 3.41. In addition, there may be a constant background illumination present in the extracted image (due to atmosphere, a defect in the camera, or some other light source) which we may want to try and capture. We can calculate this term using

$$\phi = \frac{\mu_p \mu_{p2} - \mu_p \mu_{pe}}{\mu_{p2} - \mu_p^2} \tag{3.43}$$

where ϕ is the constant background term, μ_p is the average of the illumination values of the predicted image pixels that have valid data in both the predicted and extracted images, μ_{p2} is the average of the squared illumination values of the predicted image pixels that have valid data in both the predicted and extracted images, and μ_{pe} is the average of the product of the illumination values of the predicted and extracted images at pixels where there is valid data for both the predicted and extracted images[30]. Now we need to be careful here. Since we need to use the images themselves to estimate the background term we need the predicted and extracted images to be of the same area. Therefore it is beneficial to first check the normalized cross correlation between these images and if the correlation value is low then to not try and estimate the background term (if the correlation value is low then it indicates that the predicted and extracted images do not match well and therefore are likely not of the same area). The correlation value can be computed using the steps discussed in Sec.3.5 and then checked against a user specified value to protect against this[31].

If the correlation check succeeds and we calculate a value for the background term then we need to adjust our weighting term to account for the background term. We do this using

$$\lambda = \frac{\mu_e - \phi}{\mu_p} \tag{3.44}$$

where λ is the weighting term. Therefore we can now describe our predicted image as

$$I_k = \lambda a_k \left((1-\beta)\cos i_k + \beta \frac{\cos i_k}{\cos i_k + \cos r_k} \right) + \phi \tag{3.45}$$

[28] There is also a third illumination model; however, this was developed solely for Vesta and is thus worthless for any other body

[29] There appears to be a missing factor of 2 here that is seen in the literature. The issue appears to be that the factor of 2 is missing from the abstract of McEwen's paper but is included in the body. Need to ask Dr. Gaskell about this.

[30] Figure out what is going on here and where this comes from.

[31] Since this is a normalized cross correlation the maximum correlation value is 1 which will only occur if the image is correlated with itself. Because of this we can always skip calculating the background term by setting the correlation check to a value of 1, which will never be achieved since we are working with real data.

which completes our image prediction.

3.3.2 Image extraction

The other side of the image preparation is the extraction of the maplet from the photo input into the system. This process, which is actually much more complicated than the creation of the predicted image, is completed in two main steps, with each main step having numerous smaller steps. The first step is to perform the actual extraction of the illumination data onto the maplet surface, which is completed by first determining the corresponding points between the maplet and photo and then by either interpolating (if the maplet grid size is approximately the same size or smaller than the ground sample distance of the camera pixels) or averaging (if the size of the maplet grid is much larger than the ground sample distance of the camera pixels) the illumination values and assigning these values to each maplet grid location. Once the image has been extracted onto the maplet surface, the data is "filtered" to handle shadows and other non-visible data. The filtering is done by extending shadows to make up for the fact that shadows are not sharp, by removing data that should not be visible due to the topography of the maplet using reflection angles and ray tracing, and by removing data that should be shadowed according to ray tracing.

3.3.2.1 Projection onto the maplet topography

The first step in image extraction is to project the illumination data onto the maplet topography. In order to do this we must determine the correspondences between the expected location of the landmark in the photo frame and the location of the landmark in the asteroid-fixed frame. We know the asteroid-fixed location of the landmark from our maplet and shape models, therefore we just need to calculate the expected photo frame location of this point based on our current best estimate of the spacecraft's state at the time the picture was captured. This is a straight forward mapping through the Owen camera model described in Sec. 1.1.1. Our next step is determine the photo frame locations of the rest of the maplet grid. We do this by calculating the asteroid-fixed location for each grid point in the maplet frame that has height data, and then mapping it through the Owen camera model to get the photo frame location. The asteroid-fixed location for each point in the maplet is given by

$$(\mathbf{g}_k)_B = s_m \mathbf{T}_B^M (\mathbf{x}_k)_M + \mathbf{v}_{body-lmk} \tag{3.46}$$

where $(\mathbf{g}_k)_B$ is the asteroid-fixed location of the k^{th} point in the maplet, s_m is a scaling term to make the units into kilometers, $\mathbf{T}_B^M$ is the rotation matrix from the maplet frame to the asteroid-fixed frame, $(\mathbf{x}_k)_M$ is the k^{th} point in the maplet[32], and $\mathbf{v}_{body-lmk}$ is the asteroid-fixed location of the maplet frame origin.

With the photo frame locations of the maplet points in hand we now need to check the relative size of the maplet grid compared to the picture grid. We do this by first determining the Jacobian matrix that relates a change in the maplet frame to a change in the photo frame and then taking the determinant of the x and y components of this Jacobian which gives us the number of camera pixels per maplet grid (see Eq. 3.15 and surrounding for a description of this process and explanation of what the determinant approximates). Once we have our camera pixels per maplet grid we now decide whether we are just going to use bilinear interpolation, or whether we are going to average the illumination values of the photo to determine the illumination in the maplet frame. If the camera pixels are larger than the maplet grid size then we will just use bilinear interpolation. If the maplet grid size is bigger than the camera pixels then we will sample the illumination within each pixel that the maplet grid covers and average these values to get our illumination for that maplet grid (note that we will use bilinear interpolation in the sampling of the camera pixels for this case as well).

For the bilinear interpolation, we must first determine the base pixel for each location we are trying to interpolate. We can do this by rounding the location we are trying to interpolate minus half a pixel to the

[32]Note that each maplet point will comprised of integers for x and y and height values for z.

nearest hole number[33]. With our base pixel determined, we can now define our grid for bilinear interpolation as the base pixel, the pixel to the immediate right of the base pixel, the pixel immediately below the base pixel, and the pixel immediately to the right and below of the base pixel[34]. Once we have our grid defined we can begin the interpolation. We get the coordinates for our interpolation by subtracting the base pixel from the point of interest (this should give us coordinates between 0 and 1). We can then calculate the interpolation coefficients as

$$b_0 = I_0 \tag{3.47}$$
$$b_1 = I_1 - I_0 \tag{3.48}$$
$$b_2 = I_2 - I_0 \tag{3.49}$$
$$b_3 = I_0 + I_3 - I_1 - I_2 \tag{3.50}$$

where b_i are the interpolation coefficients, and I_i are the illumination values at the corners of the interpolation grid, 0 corresponding to the base grid location and 3 corresponding to the lower right grid location. Finally, our interpolated value is

$$i = b_0 + b_1 g_x + b_2 g_y + b_3 g_x g_y \tag{3.51}$$

where i is the interpolated illumination value at our point of interest and g_x and g_y are the coordinates of our point of interest.

3.3.2.2 Extract filter

Just determining the illumination values on the surface of the maplet is usually not a good idea. This is because the maplet is not flat. Therefore, just because interpolation gives an illumination value for a certain point on the maplet, it doesn't mean that the point should actually be illuminated or visible according to the current best estimate of the spacecraft's state. For instance, imagine a point located on the side of a mountain opposite the current best estimate of the spacecraft. Because of misalignment and imaging noise, our bilinear interpolation is likely to find an illumination value for this point on the maplet. In real life, however, we know that there is no chance we can see anything on that side of the mountain, therefore it would be best to zero that data out and to not include it in our correlation. These types of checks are handled by the extract filter.

The extract filter relies heavily on two vectors to performs its functions. They are the incidence vector and the reflection vector. The incidence vector is the unit vector that points from the body to the sun expressed in the maplet frame (this assumes collimated light). The reflection vector is the unit vector that points from the maplet frame to the spacecraft expressed in the maplet frame.

The first thing the extract filter does is to "enhance" shadows. Enhancing shadows means that it extends any shadows to account for the fact that shadows are not sharp (according to the SPC source code at least). To enhance the shadows, we first must consider in what direction the shadows are propagating. This is done by considering the projection of the incidence vector onto the maplet frame $xy-$plane. Once we have the direction the shadows propagate in we need to locate the edges of the shadows in the extracted image. In the SPC software, shadows and invisible data are represented by illumination values of 0. Therefore, we need to search for any grid cells in the extracted image that are nonzero and the three pixels along the direction of the incidence vector are 0.

[33]To understand why we do this consider how the image pixel works. Each row and column corresponds to a pixel. Each pixel has a size of 1×1 in units of pixels. Therefore the boundaries between each pixel are located at $+0.5$ and -0.5. Therefore, when we round to the nearest hole number we will be getting the pixel that contains each point we are interested in. By first subtracting half a pixel from each point we are interested in we actually adjusting so that any time we are interested in a point that is in the -0.5 half of the pixel our pixel of interest will actually become the previous pixel. This is important because when we are defining our corners for bilinear interpolation, we want to ensure that the point of interest is contained within the box formed by those corners. If we didn't adjust, then when we had a point of interest that was in the -0.5 half of a pixel our corners would start at the nearest pixel to this point, and move to the right and down, and thus would not include our point of interest.

[34]In the SPC source code there are additional checks to see if there is valid data at these points, and if there is not to try to get valid data from other surrounding pixels but this is not discussed here as it is pretty much just a long string of conditional statements

Once we have identified the edges of the shadows in our image we can begin our enhancements. The first thing to determine is how much we should extend our shadows. This is done by considering the relative resolutions of the maplet and image frames. If the maplet grid is at a finer resolution than the picture, then we want to extend our shadows even more, if the picture is a finer resolution then we only want to extend our shadows by some base amount. This can be figured out again using the determinant of the Jacobian from the maplet frame to the photo frame:

$$ds = 3 + \left\lfloor \frac{1}{\det \mathbf{J}_{xy}} \right\rfloor \tag{3.52}$$

where ds is the number of grid cells to step along, 3 is the base extension of the shadows, $\mathbf{J}_{xy}$ is the x and y columns of the Jacobian matrix from the maplet frame to the photo frame, $\det \bullet$ indicates the determinant of a matrix, and $\lfloor \bullet \rfloor$ indicates that the floor of the value should be taken (round down to the closest integer).

Now with our step size we can extend the shadows. Each pixel in the direction opposite of the incidence vector projected onto the maplet frame within a distance of ds is zeroed out. Further, to help with blurry images, we will also zero out cells that are along the inverse incidence vector within plus or minus two of these points[35]. This concludes the enhancement of the shadows.

The final function performed by the extract filter is to zero out data that should be invisible according to the geometry of the scene. This is done by placing a restriction on the reflection angle as well as through ray tracing. In order to zero out data according to the reflection angle, we simply need to take the dot product of the reflection vector with the local normal vectors (as described by Eq. 3.37). If the dot product between the reflection vector and the local normal vector is either negative or very small then it means that point on the surface is being viewed either from behind (if the dot product is negative) or at a very sharp oblique angle (if the dot product is very small) and thus we cannot really see this patch of surface and should zero out the data in the containing grid cell.

In order to remove invisible data through ray tracing, all we need to do is step along the reflection vector from each cell in the maplet and check to see if the heights of the maplet ever obscures this vector. For each point in the maplet the reflection vector is stepped along at a distance of one cell length intervals. At each step along the reflection vector, the height of the maplet at that x and y point is interpolated using bilinear interpolation. Finally, the interpolated height is compared with the current height of the step point and if the interpolated height is higher then the originating cell is marked as invisible and zeroed out[36].

This concludes the extract filter and the preparation of the images[37].

3.4 Removing Maplets

After we have added the maplets that are visible in the image we may have found many maplets. This can be undesirable because it can take a long time to process a lot of maplet per image. In addition, it is likely that there are a good number of maplets that are not actually very useful due to their illumination conditions or orientation with respect to the camera frame. Therefore, it may be beneficial to enforce stricter conditions on the maplet geometry and illumination conditions. We can do this by either checking against five additional criteria on the maplets or by manually specifying which maplets to remove (usually done by visual inspection of the illuminated maplet image versus the extracted maplet image).

For the automatic removal of landmarks we need to check the removed data ratio of the extracted maplet images (sometimes referred to as the invisible ratio of the maplet, but this can lead to confusion with the

[35] I don't really know what is happening in the code here. For some reason we also zero out data that is located $\pm 2[y_{inc}\ x_{inc}]^T$ but there is no explanation as to why...

[36] In the SPC code the reflection vector is artificially lowered by a small tolerance, likely to attempt to avoid allowing cells that are partially occluded to remain illuminated

[37] In the SPC source code, the extract filter claims that it also uses ray tracing to zero out data that should be shadowed. While there is code present to do this it is currently skipped over by a "goto" statement and there is no chance the code will ever be actually run. I have ignored this code in this paper since it is not used but it acts exactly the same as the ray tracing for the invisible data, except instead of stepping along the reflection vector we step along the incidence vector.

invisible checking discussed before), the reflection angle (referred to as the emission angle in the SPC source code, although emissions come from black bodies not illuminated bodies), the illumination percentage of the extracted maplet images (referred to as the coverage percentage or the maplet-photo overlap in the SPC source code), and the relative resolution of the photo pixels to the maplet grid size. The removed data ratio of the extracted maplet images is simply the ratio of the number of pixels that were zeroed out by the extract filter to the total number of pixels in the extracted image plus one. That is

$$data_r = \frac{rem_{pix}}{N^2 + 1} \tag{3.53}$$

where rem_{pix} are the pixels that were set to 0 by the extract filter and N is the total number of pixels along one side of the predicted images ($(2 * qsz + 1)$ in the SPC source code). The reflection angle check just checks to be sure that the reflection angle is not too large[38]. We can calculate the reflection angle as

$$r = \cos^{-1}(\hat{\mathbf{r}}_{Mz}) \tag{3.54}$$

where r is the reflection angle and $\hat{\mathbf{r}}_{Mz}$ is the $z-$component of the line-of-sight unit vector from the maplet to the camera center expressed in the maplet frame[39]. The illumination percentage of the extracted maplet image is simply the ratio of the total number of image pixels that have positive illumination values over the total number of pixels. That is

$$illum_r = \frac{\sum_{i,j}^{N} \mathbf{I}_p(i,j) > 0}{N^2} \tag{3.55}$$

where $illum_r$ is the illumination ratio, $\mathbf{I}_P(i,j)$ is the illumination value of the predicted image at row i and column j, and summing over a logical equation means to add a 1 if the equation is true and to add a 0 if the equation is false. Finally the relative resolution between the maplet and image can be calculated as

$$res_r = \frac{r_{img}}{s_m} \tag{3.56}$$

where res_r is the resolution ration between the maplet cells and the photo pixels, r_{img} is the resolution of the photo at the maplet in units of kilometers (or some other distance) per photo pixel, and s_m is the maplet scale factor with units of kilometers (or some other distance) per maplet cell[40].

For each of these values, we compare it to a value specified by the user and throw out any maplets that do not meet the requirements. For the good data ratio, if the computed value is less than the user specified value then the maplet is kept (note that the user specified value should be the desired ratio times 1000 rounded to the nearest hole number). For the check on the reflection angle, maplets with reflection angles lower than the user specified value are kept (note that the user value should be entered in degrees). For the check on the illumination ratio, maplets where the computed value is greater than or equal to the user specified value are kept (the user value should be entered in terms of the actual ratio). Finally two checks are performed on the resolution ratio. The first check is that the resolution ration must be less than some user specified maximum value and the second check is that the resolution ratio must be greater than some user specified minimum value for the maplet to be kept (both values should be entered in terms of the actual ratios desired).

3.5 Correlation

With the images prepared we can now turn to determining the offset that best aligns them. This can be done through the use of a 2D cross correlation. A 2D cross correlation is the dot product between the two images,

[38] Recall that the reflection angle is the angle between maplet frame $z-$axis and the line-of-sight vector from the camera to the maplet, therefore if this is large it means the maplet is being viewed at a large angle. Also remember that we have already checked this value when we added the maplet so this is just a chance to put a stronger requirement on this angle.

[39] This is just a simplification of the dot product formula $\mathbf{x}^T\mathbf{y}^T = \|\mathbf{x}\|\|\mathbf{y}\|\cos(\theta)$ taking into account that the maplet frame $z-$axis expressed in the maplet frame is $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ and the fact that both $\hat{\mathbf{r}}_M$ and the maplet frame $z-$axis are both unit norm.

[40] A little more discussion is warranted here because this value can often be a point of confusion. This value has units of maplet cells over photo pixels. Therefore, if this ratio is greater than 1 then it means that there is more than one maplet cell per photo pixel, and thus the maplet has a higher resolution (each cell covers a smaller distance) then the photo. If the ratio is less than 1 then it means that each maplet cell covers more than 1 photo pixel and thus the photo is higher resolution than the maplet.

that is, you take each overlapping element and multiply them together, and then sum all of these products. We can do this many times, changing how the two images are overlapped each time, in order to generate a correlation surface of the correlation values at each layout of the images. Once we get the correlation surface we can use numerous techniques to try and get the sub-pixel offset that would provide the best correlation.

In SPC the correlation is a little more complicated than what is described above. First, it is actually a normalized cross correlation, which makes it easier to compare the correlation peak values. Second, in order to save memory and time the bounds of the correlation are restricted. This means that to attempt to correlate over larger distances than what is in the bound we must re-sample the images to a lower resolution. We will now begin discussing exactly what is happening in the SPC correlator.

In the SPC correlator, the predicted image is moved over-top of the extracted image. This is done in increments of full pixels (so that we don't have to perform bilinear interpolation each new offset we try in the correlator). Further, as mentioned above, if we are trying to correlate over anything greater than a 11×11 grid then the images are binned to a lower resolution instead of increasing the step size. This helps to keep the amount of computation required for the correlator constant even when large offsets are required (although after performing a binned correlation it is recommended to eventually perform a non-binned correlation by updating the images using the binned correlation result and decreasing the size of the binning by a factor of 1 each time). The binning process is fairly straight forward, it is a simple averaging binning. If we wish to correlate across a range of $5n$ pixels in both the row and column directions of the images, then we bin the images as

$$\mathbf{I}_b(r_b, c_b) = \frac{\sum_{i=1+(r_b-1)n}^{(r_b-1)n} \sum_{j=1+(c_b-1)n}^{(c_b-1)n} \mathbf{I}(i,j)}{\sum_{i=1+(r_b-1)n}^{(r_b-1)n} \sum_{j=1+(c_b-1)n}^{(c_b-1)n} \mathbf{I}(i,j) > 0} \tag{3.57}$$

where $\mathbf{I}_b(r_b, c_b)$ is the illumination value of the binned image at pixel (r_b, c_b), $\mathbf{I}(i,j)$ is the illumination value of the full image at pixel (i,j), n is the multiplier specifying the number of pixels we want to correlate over, and the summation on the logical equation means to add a 1 if true and a 0 if false. We then correlate over a 11×11 grid on the binned images, effectively correlating over the $11n \times 11n$ region in the original images we desired. Note that this is a courser correlation than when $n = 1$. This is because the size of the pixels we are correlating over are actually the size of n^2 pixels in the original images, and thus, when we determine our peak correlation location to some sub-pixel accuracy, it is sub-pixel accuracy in the new larger pixels, and thus likely not sub-pixel accuracy for the true pixels.

With the binned images in hand we can turn to the task of correlating them over our 11×11 grid. As mentioned before, the correlator used is a normalized correlation technique. This means that our correlation values are constrained to fall between -1 and 1. In addition it means that we are correlating based on unbiased images (zero mean) with variances normalized to be equal to 1. In the SPC source, the correlation for each step is performed as

$$C = \frac{\frac{1}{N} \sum_{i=1}^{k} \sum_{j=1}^{l} \left(\mathbf{I}_{pb}(i,j)\mathbf{I}_{eb}(i,j)\right) - \mu_{pb}\mu_{eb}}{(\mu_{pb2} - \mu_{pb}^2)(\mu_{eb2} - \mu_{eb}^2)} \tag{3.58}$$

where C is the correlation value, N is the number of pixels that are nonzero in both the binned predicted (offset) and extracted images, μ_{pb} is the average illumination value of the binned and offset predicted pixels that are nonzero in both images, μ_{eb} is the average illumination value of the binned extracted pixels that are nonzero in both images, μ_{pb2} is the average of the square of the illumination values of the binned and offset predicted pixels that are nonzero in both images, and μ_{eb2} is the average of the square of the illumination values of the binned extracted pixels that are nonzero in both images. The correlation value is calculated over a 11×11 grid, where the predicted image has been offset from the center by some amount contained in $-5n : n : 5n$ pixels and then binned.

It may be beneficial to now discuss where this formula for the normalized crossed correlation comes from[41]. In the standard normalized cross correlation we take the dot product of two zero mean unit standard deviation images. That is

$$C = \frac{\sum \left(\mathbf{I}_{pb} - \mu_{pb}\right)\left(\mathbf{I}_{eb} - \mu_{eb}\right)}{\sigma_{pb}\sigma_{eb}} \tag{3.59}$$

where μ_{pb} is the average illumination value of all pixels in the binned predicted image that are nonzero in both binned images, μ_{eb} is the average illumination value of all pixels in the binned extracted image that are nonzero in both binned images, the summation sign means to sum over all pixels that have valid data in both images (we will now use this notation to make the following derivation more efficient), and $\sigma_\bullet$ indicates the standard deviation of all of the elements of the matrix (or all of the pixels with valid data in the images)[6]. This is obviously different from Eq. 3.58 and in fact, if we simply expand the multiplications and substitute in the definition of a standard deviation then Eq. 3.58 looks wrong

$$C = \frac{\sum \left(\mathbf{I}_{pb}\mathbf{I}_{eb} - \mathbf{I}_{pb}\mu_{eb} - \mathbf{I}_{eb}\mu_{pb} - \mu_{pb}\mu_{eb}\right)}{\sqrt{\sum\left(\mathbf{I}_{pb}^2 - 2\mu_{pb}\mathbf{I}_{pb} + \mu_{pb}^2\right)\sum\left(\mathbf{I}_{eb}^2 - 2\mu_{eb}\mathbf{I}_{eb} + \mu_{eb}^2\right)}} \tag{3.60}$$

as it appears like we have lost the cross terms of the product in the numerator as well as in the equations for the standard deviation[42]. If we further expand this we can see what happened. Substituting in the definitions of the norms and distributing the sums we get

$$C = \frac{\sum \mathbf{I}_{pb}\mathbf{I}_{eb} - \sum \mathbf{I}_{pb}\sum \mathbf{I}_{eb}/N - \sum \mathbf{I}_{eb}\sum \mathbf{I}_{pb}/N - N\sum \mathbf{I}_{eb}\sum \mathbf{I}_{pb}/N^2}{\sqrt{\left(\sum \mathbf{I}_{pb}^2 - \frac{2\sum \mathbf{I}_{pb}\sum \mathbf{I}_{pb}}{N} + \frac{N\sum \mathbf{I}_{pb}\sum \mathbf{I}_{pb}}{N^2}\right)\left(\sum \mathbf{I}_{eb}^2 - \frac{2\sum \mathbf{I}_{eb}\sum \mathbf{I}_{eb}}{N} + \frac{N\sum \mathbf{I}_{eb}\sum \mathbf{I}_{eb}}{N^2}\right)}} \tag{3.61}$$

from which we can simplify to get

$$C = \frac{\sum \mathbf{I}_{pb}\mathbf{I}_{eb} - \sum \mathbf{I}_{pb}\sum \mathbf{I}_{eb}/N}{\sqrt{\left(\sum \mathbf{I}_{pb}^2 - \sum \mathbf{I}_{pb}\sum \mathbf{I}_{pb}/N\right)\left(\sum \mathbf{I}_{eb}^2 - \sum \mathbf{I}_{eb}\sum \mathbf{I}_{eb}/N\right)}} \tag{3.62}$$

Now, if we simply multiply by the unitary $(1/N)/(1/N)$ then we can get back to the form of Eq. 3.58 which completes the derivation of the SPC correlator.

3.5.1 Sub-pixel detection

Once we have correlated our images for every offset we can now attempt to extract out the sub-pixel location of the peak correlation value. We do this by first locating the pixel location of the peak correlation value (which is just the maximum correlation value). Taking the pixel level shift that created the highest offset we can get sub-pixel accuracy by considering the surrounding correlation surface. We do this by fitting a curve to the surface and locating the peak location of the curve, which is also presumably the sub-pixel offset that would give us the absolute highest correlation peak.

There are many types of curves and methods we could use to get the sub-pixel accuracy. In the SPC source code a relatively simple technique of fitting a first order Taylor series to the gradient of the correlation surface is used. Furthermore, the SPC source assumes that the function governing the correlation surface is independent in the row and column directions; therefore, we can perform 2 first order 1D fits to the correlation surface rather than performing a first order 2D fit. The Taylor series fit used for each direction is

$$\frac{\mathrm{d}y}{\mathrm{d}x}(x) \approx \frac{\mathrm{d}y}{\mathrm{d}x}(x_0) + \frac{\mathrm{d}^2y}{\mathrm{d}x^2}(x_0)\left(x - x_0\right) + \ldots \tag{3.63}$$

[41] I spent numerous hours trying to figure out how to get from the standard normalized cross correlator to the one present in the SPC code. Therefore, I would imagine other people might be similarly confused. In addition I want to give purpose to my needless struggle of not thinking to use simple algebra ...

[42] Note that the square terms in the denominator mean to square each element of the matrix, not to multiply the matrix by itself. Yes this notation sucks but believe me when I say that it is much more readable than the more precise notation we were using before

where we have fit a Taylor series expansion to the derivative $\mathrm{d}y/\mathrm{d}x$ about point x_0 with y representing the correlation values and x representing a change in either the row or the column direction of the correlation surface. We will do this fit in both the row and column direction. Now what we really want out of these fits is to solve for the row and column that gives the peak value. Therefore we can rearrange our Taylor series expansion to be

$$x \approx x_0 + \frac{\mathrm{d}^2 x}{\mathrm{d}y^2}(x_0)\left(\frac{\mathrm{d}y}{\mathrm{d}x}(x) - \frac{\mathrm{d}y}{\mathrm{d}x}(x_0)\right) \tag{3.64}$$

Further, since we know we are looking for the peak value, assuming the correlation surface is a smooth function, we know that the peak value must occur when the first derivative equals 0 by the first derivative test[43]. Therefore, we can solve for the offset that we desire by setting the $\mathrm{d}y/\mathrm{d}x(x)$ term equal to 0. All that remains is to calculate the derivatives at the pixel level peak correlation value which can be done through finite differencing:

$$\frac{\mathrm{d}^2 x}{\mathrm{d}y^2}(x_0) = \left(y(x_0+1) + y(x_0-1) + 2y(x_0)\right)^{-1} \tag{3.65}$$

$$\frac{\mathrm{d}y}{\mathrm{d}x}(x_0) = \left(y(x_0+1) - y(x_0-1)\right)/2 \tag{3.66}$$

where $x_0 \pm 1$ indicates a step right/left or up/down in the correlation surface, depending on whether we are considering the horizontal or vertical fit. This completes the sub-pixel fit of the correlation surface.

3.5.2 Maplet frame to image frame conversion

All that remains in the SPC routines is to transform the peak correlation location into the location of the maplet in the image frame. We do this by first multiplying the peak correlation offset from the middle of the correlation surface by the step size that we used while correlating. This gives us the offset between the predicted location of the landmark and the actual location of the landmark in terms of maplet image pixels. Now we need to convert these maplet pixels into the pixels of the photo frame. This is done by once again turning to the Jacobian that transforms from the maplet frame to the photo frame. All we need to do is multiply the shift expressed in terms of maplet pixels by the first two columns of the Jacobian matrix, that is

$$\mathbf{\Delta l}_I = \mathbf{J}_{xy}\mathbf{\Delta l}_M \tag{3.67}$$

where $\mathbf{\Delta l}_I$ is the shift between the expected and actual landmark locations in the image frame and $\mathbf{\Delta l}_M$ is the offset of the peak correlation value in the maplet frame. This completes the SPC for navigation processing. The results can now either be used to update the spacecrafts state, or used to re-extract and predict the maplet images so that a better correlation can be achieved.

4 Conclusion

In this document we have presented the basic mathematical ideas behind the OSIRIS-REx surface feature OPNAV software. We began by introducing the ideas of surface feature OPNAV and explaining the frames and frame transformations that are used throughout the software. We then explained the software that uses the 2D-3D point correspondences generated by SPC for navigation to update the state of the spacecraft. Finally we examined each module of the SPC for navigation routines and explained how they are implemented.

[43] Yes the first derivative test only guarantees a local extrema; however, there are checks in the code that require that the correlation values are all decreasing around the correlation peak (thus the correlation peak cannot be on the edge of the correlation surface) and this ensures that the first derivative test will actually identify only a maximum...

5 Acknowledgments

As mentioned above, this entire document is based off of the theory and code developed by Dr. Gaskell, and thus none of this would have been possible without his work. Further, I would like to thank Kenneth Getzandanner for his help in reviewing this manuscript and figuring out exactly what was happening in the SPC source code. In addition I would like to thank him for helping to get me a job at Goddard because otherwise I would not have created this document.

References

[1] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2003.

[2] William M Owen Jr. Methods of optical navigation. *Spaceflight Mechanics*, 140, 2011.

[3] Juyang Weng, Paul Cohen, and Marc Herniou. Camera calibration with distortion models and accuracy evaluation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (10):965–980, 1992.

[4] Alfred S. McEwen. Photometric functions for photoclinometry and other applications. *Icarus*, 92(2):298 – 311, 1991.

[5] RW Gaskell, OS BARNOUIN-JHA, DJ Scheeres, AS Konopliv, T Mukai, S Abe, J Saito, M Ishiguro, T Kubota, T Hashimoto, et al. Characterizing and navigating small bodies with imaging data. *Meteoritics & Planetary Science*, 43(6):1049–1061, 2008.

[6] JP Lewis. Fast normalized cross-correlation. In *Vision interface*, volume 10, pages 120–123, 1995.