ON-BOARD LIMB-BASED SHAPE MODELING FOR SMALL BODY NAVIGATION Josh R Lyzhoft[1], D. Baker[1], A. Liounis[1], M. Fickett[1], J. Swenson, [1]NASA Goddard Space Flight Cetner, Greenbelt, 8800 Greenbelt Rd 20771 USA; joshua.r.lyzhoft@nasa.gov

Keywords: *Limb-based, Shape modeling, On-board*

Abstract: *Missions to small bodies within our solar system are becoming more frequent. Generally, shape models of the target body are required to perform proximity operations as demonstrated by the OSIRIS-REx, Hayabusa2, and Rosetta missions. However, these missions required image downlinking to create high-resolution models on the ground. In some missions, especially smallsats, there may be data downlink data constraints, resulting in the inability to provide the large number of images needed for high-resolution shape models. A solution to this is the ability to generate shape models on-board during the approach to the target or initial proximity surveying. Current work implements a limb-based shape model routine that is able to be executed on a Raspberry Pi 1, which is similar to the processing capability to the flight computer on OSIRIS-REx, and that does not require low phase angle geometries. Initial shape model results generated from Bennu approach (Nov 2-3, 2018) show that the limb-based shape model agrees well with a 75-cm SPC shape model generated after Preliminary Survey; Results are: min difference -7.047m, max difference 17.347m, mean 1.122m, and RMS 3.219m. Further scenarios are presented herein.*

Introduction: Missions to small bodies within our solar system are becoming more frequent due to the decreasing overall cost. Recent examples of small body missions are: Rosetta to comet 67P/Churyumov-Gerasimenko, Hayabusa2 to Ryugu, and OSIRIS-REx to Bennu. Each one of these missions conducted proximity operations around their respective bodies, which required high resolution shape models.

The Rosetta mission used at least two ways of shape modeling: stereophotoclinometric (SPC) and stereophotogrammetric (SPG).[1] Both of these methods required high resolution images to be downlinked to Earth. Downlink bandwidth was not a major constraint, generally, due to the priority of the mission.

The Hayabusa2 spacecraft's tour around Ryugu consisted of shape modeling, firing a projectile into the surface, and collecting a sample. Modeling of the asteroid was done using two methods: Structure-from-Motion (SfM) and SPC.[2] Again, downlinks are required. However, conducting a sample return required the amount of accuracy and precision that these methods provide.

OSIRIS-REx performed a sample collection of the surface of asteroid Bennu.[3] Initially, the mission had a preliminary radar shape model that was used for Approach, which was slightly off from the proximity estimated models.[4] After obtaining imaging from PolyCam, MapCam, and NavCam 1 during the Approach and Preliminary Survey phases, a 75-cm resolution SPC shape model was generated.[5] Other models based on LiDAR measurements and higher resolution images were also created after phases later in the mission. Just like the previous missions, a high resolution shape model was required and could not be generated on-board the spacecraft.

Methods that do not require such a computationally intense estimation routine have been formulated before. Such methods use some basics of shape from silhouette. Volume carving generates models using an iterative method to slowly refine the volume an object occupies by obtaining silhouettes from different viewing geometries.[6,7] These particular methods have not been done on space mission targets as of yet. This is due to the requirement of near-zero phase angle resulting in complete illumination of the observed object from an image's viewpoint.

A similar concept to shape from silhouette, limb-based shape modeling, was proposed by Baker.[8] Here, since an object is floating in the abyss of space, the limb of the target is determined. Then a 3D representation of the shape is generated by determining "patches" that are contained with intersecting planes. This can be computationally intensive due to the plane intersection determination and patch search.

Proposed Method: This section seeks to elaborate on an on-board limb-based method for creating a shape model of a celestial body. The shape model is constructed using various images of a target, viewing different portions of its limb.

Overview: The current research involves limb-based modeling as discussed in 8. However, a different approach for rendering the 3D model is conducted. Instead of searching for intersecting planes and patches, this method generates potential vertex points towards and away from the camera at each determined limb point in each image.

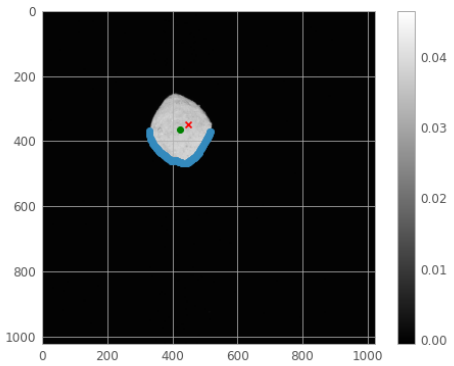


Figure 1: Part of PolyCam image sequence (1 image out of 222 taken from Nov 2, 2018 to Nov 3, 2018) taken during Approach. Estimate limbs (blue), projected center-of-mass from trajectory files (red), and estimated 2D center-of-figure (green) corrected due to phase angle

Figure 1 depicts the limb's points found for an image along with estimated 2D center-of-figure and center-of-mass given by trajectory files. Each point is then projected onto a plane that is created by the body's orientation with respect to the viewing geometry. Thereafter, for each image, a polygon is created by using the limb points, and the points are trimmed by using a point in polygon routine. After the points have been trimmed and outward-pointing normal vectors estimated, the surface is reconstructed using Poisson surface reconstruction.[9] A general process of the algorithm flow can be seen in Figure 2

A few advantages over previous methods are the proposed algorithm: 1) reduces the computational dimensions by projections, 2) uses phase angle and target pixel size estimations to incorporate a correction to the center-of-mass calculation, 3) has the ability to update a shape model with newly taken images, and 4) is able to be executed on a Raspberry Pi 1.

Assumptions: A few requirements are needed in the formulation of this limb-based model scheme: the body-fixed frame of the target with respect to camera frame, the relative state of the target with respect to the camera, the camera pointing information for each image including image meta data, and the camera information. However, as seen in Figure 1, the center-of-mass for a target is not always well represented in reconstructed trajectory files, predicted trajectory files, or camera pointing information. For convex shapes, the phase angle-corrected centroid can be used for the center-of-mass.

Limb Point Determination: Limb points are generated by projecting the incoming sun vector onto the image plane and then shooting rays parallel to the sun vector. If a ray encounters the target, based on a gradient, a limb point is placed where this larger gradient occurs, and a subpixel routine is used to better estimate the limb point location. Figure 3 shows the general idea of limb point determination and limb-based polygon creation, which will be discussed in later in the document.

For the surface reconstruction method used herein, each surface vertex must have an outward pointing normal vector. Outward-pointing normal vectors at each vertex can be approximated by using the average of each line segment on either side of a vertex. This is done by first rotating the limb points in order for the sunlight to come directly from the positive x-direction. Thereafter, the limb points can be reordered in y-value ascending order. Doing so allows for easier computations of the line segment outward-pointing.

The line segment outward-pointing normal vectors are computed by:

$$\vec{l}_i = [y_i - y_{i-1}, x_{i-1} - x_i] \tag{1}$$

$$\hat{l}_i = \frac{\vec{l}_i}{||\vec{l}_i||_2} \tag{2}$$

where $\vec{l}_i$ is the ith outward vector, y_i is the y-component of the rotated and reordered ith limb, x_i is the x-component of the rotated and reordered ith limb, $i-1$ is the previous limb point, and $\hat{l}_i$ is the ith normalized outward-pointing line segment unit vector. With the each line segnment outward vector known, the vertex outward vector can be estimated as:

$$\vec{n}_{i-1} = \vec{l}_i + \vec{l}_{i-1} \tag{3}$$

$$\hat{n}_{i-1} = \frac{\vec{n}_{i-1}}{||\vec{n}_{i-1}||_2} \tag{4}$$

Due to the number of line segments being less than the number of vertexes and the limb endpoints not having a defined outward-pointing normal vector, two additional line segment outward normal pointing vectors are created. Each one of the newly added line segments are pointing in opposite direction, solely in the y-axis. Figure 4 gives a visual representation of the outward-pointing nor-

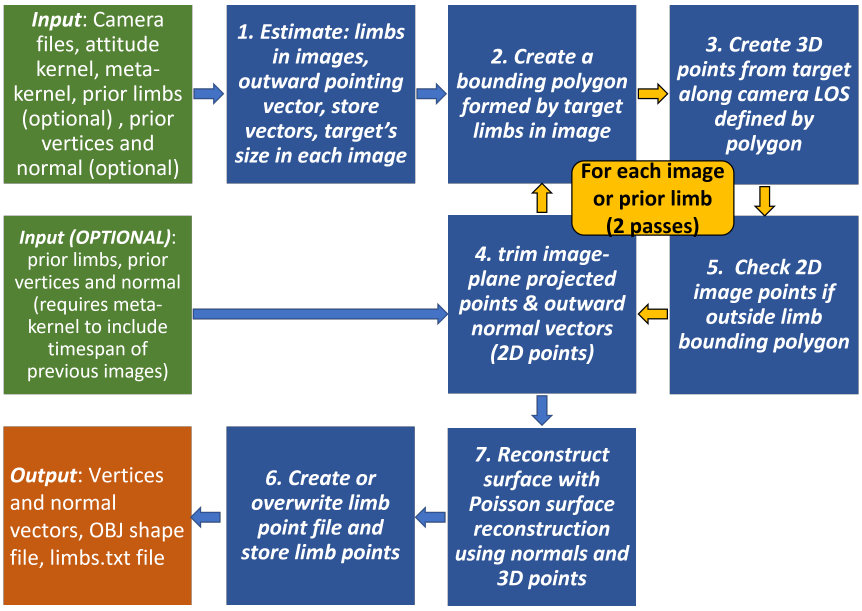


Figure 2: General work flow of the limb-based algorithm

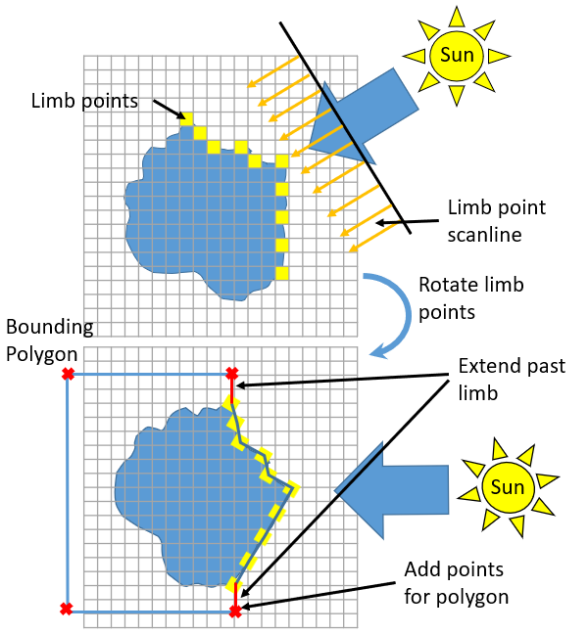


Figure 3: Depiction of limb identification and bounding polygon creation.

mal vectors for each line segment. The general flow of the algorithm can be found in Algorithm 1.

Center-of-Mass Approximation: The estimate of the center-of-mass is essential to properly place the surface vertex points in the body fixed frame. Each image requires a process to determine the center-of-mass due to the possibility of camera

Create new line segment outward-pointing normal ([0,-1] direction)
for *loop through ordered limb points starting at the second index* **do**
  Get line segment outward normal, $\hat{l}_i$;
  Estimate limb point $i-1$ outward-pointing normal unit vector, $\hat{n}_{i-1}$;
end
Create new line segment outward-pointing normal ([0,1] direction)
Compute outward-pointing unit vector for last limb using new line segment normal and last line segment normal created by the last two limb points

Algorithm 1: Algorithm flow for determining vertex outward-pointing normal vectors

pointing knowledge error or relative state error. One simple approach to approximate the center-of-mass within in an image is to perform an Otsu threshold to the image, calculate the centroid of the binary image, and perform a centroid correction using the limb points as well as phase angle. This correction when r can be calculated as:

$$\bar{c} = \left(\frac{8\left(max\left(\boldsymbol{P}_x\right) - min\left(\boldsymbol{P}_x\right)\right)}{3\pi} \right) sin^2\left(\frac{\phi}{2}\right) \quad (5)$$

where $\bar{c}$ is the correction to be subtracted from the blob centroid, $\boldsymbol{P}_x$ is the list of x components in

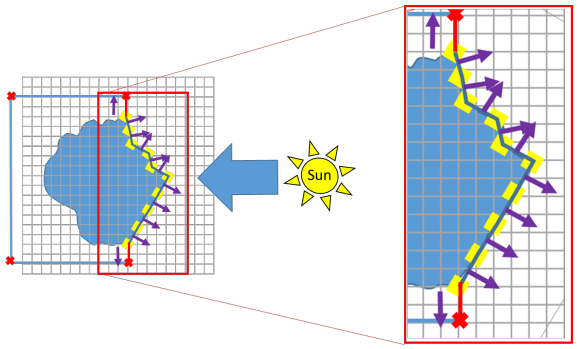


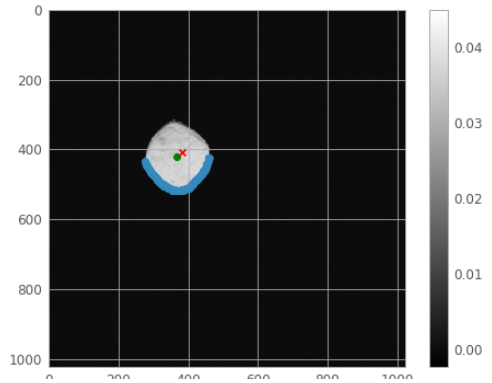
Figure 4: Depiction of outward-pointing normal vectors of each line segment connecting the limb point.

the rotated and sorted list of limb points and ϕ is the solar phase angle. However, this expression is valid for spheres and ellipsoids, when the incoming light projection is aligned with an ellipsoid's axis. The minimum and maximum values can be reduced down to an approximation of the object's diameter. There may be a small error when using the maximum and minimum formulation due to the binary image threshold and phase angle, causing an incorrect representation of target's terminator. A result of this error will cause a shift to the approximated center-of-mass towards the light direction.

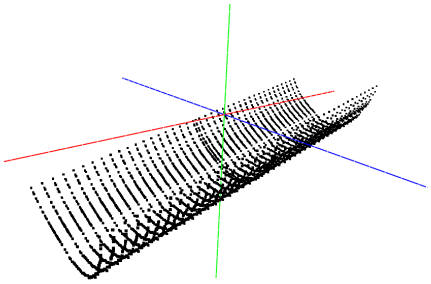
This expression has only been tested on Bennu, which is near to a convex shape. Other approximations can be done using ellipsoid fitting and phase angles. However, at this time, they have not been tested.

Three Dimensional Point Generation: Generally, convex shapes only require surface point creation at the limb point locations placed at the distance to the center of the target from the observer. When there are irregularities in the shape that may not initially be known, limb points may correspond to surface points not directly located purely in the plane perpendicular to the line-of-sight of the target. These limb points may lie on a surface that is either closer or further away from the observer. Figure 5 illustrates the potential surface point generation for a single image.

Generation of the three-dimensional points can be done by looping through discrete distances. Extrapolating these points along their pixel line-of-sight vectors helps to create a chance of capturing the potential irregularities of a shape. However, generating these extra points brings in the necessity to trim points that may be not be contained on the surface. Additionally, having a non-continuous limb surface may cause some surface information to be lost. Having the discrete points allows for rapid point-in-polygon trimming techniques instead complex surface intersection checking.



(a) Image with limbs (blue), ephemeris center (red), and estimated center (green)



(b) 3D generated points of the image in (a)

Figure 5: Generation of vertexes at each limb point, extrapolated towards and away from the camera at the target's estimated center-of-mass

Point Trimming: Point trimming is a technique that removes points from a collection vertexes. This can be done from points that have any number of dimensions. However, the computational complexity of removing points usually increases with the number of dimensions. The method for trimming points used reduces a three-dimensional vertex to a two-dimensional vertex for a given set of limb points. Point trimming is done by projecting the current three-dimensional points onto an image plane and determining if they fall outside of a polygon constructed using limb points. This process is done for each image and its corresponding limb point-generated polygon. A depiction of a bounding polygon can be see in the lower half of Figure

3. Further flow of the algorithm can be viewed in Algorithm 2.

```
Starting with a set of 3D vertex points
for loop through new images do
    Project 3D points on the image plane;
    Use the image's limb point polygon to
     perform a point in polygon check;
    if points are outside of limb point polygon
     then
        Delete vertex point from the 3D vertex
         set;
    end
end
Using the newly generate 3D vertex points
 from current run;
for loop through previous limbs do
    Project 3D points on the image plane;
    Use the image's limb point polygon to
     perform a point in polygon check;
    if points are outside of limb point polygon
     then
        Delete vertex point from the 3D vertex
         set;
    end
end
```

Algorithm 2: Algorithm flow for trimming surface vertex points

Surface Reconstruction: Surface reconstruction requires two steps. The first step is to remove any initial outlier points using a sphere that is based on the overall estimated size of the target and to add in vertex points on the surface of the sphere in areas of the vertex model that do not have vertexes. This process is done to help create closed shape when conducting the Poisson Surface reconstruction. An example of the sphere point removal and addition can be seen in Figure 6.

Poisson Surface reconstruction is the second step, which requires vertex points and their outward pointing normal unit vectors to create a surface mesh consisting of triangular facets. Another method that can be used is the Ball Pivot Algorithm, which does not require the outward normal unit vectors.[10] However, this algorithm requires the size of a ”ball” that rolls over the vertex points to create a closed surface.

Shape Model Update: Within the overall process, there is an option to update a model using different images. This allows for the shape to be generated using batches of images instead of processing all of them at once. However, if this option is desired, the limb points for each image, 3D surface vertexes and estimated 3D outward-pointing normal vectors for each vertex must be stored. Fortunately, the software stores this information after each execution of the software. Generally, these files do not occupy a large space in storage. Depending on the number of vertexes determined by the software (1000 vertexes starting), file sizes could start from 1 MB. These values are approximate and are not fully encompassing. However, the processing resources can depend on the total number of images that are being processed at once. For example, a shape estimation using 60 images used approximately 180 MB (120 MB from images). Further access speeds and reduced storage can be achieved by writing output binary files or loading one image at a time.

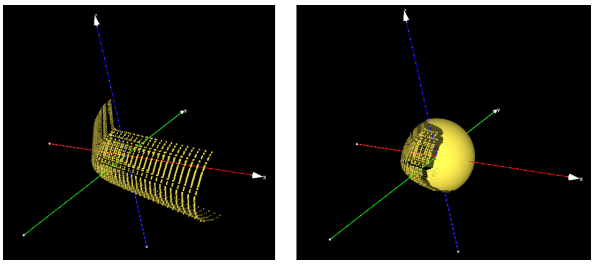


Figure 6: Sphere point removal and vertex addition after using 15 images. Left: vertexes prior to sphere removal and addition. Right: vertexes after sphere removal and addition

Preliminary Results: This section describes the different scenarios that were tested with the limb-based shape model method developed. Each setup uses trajectory files, attitude files, and images corresponding the specific mission. The missions of interest to be studied are: OSIRIS-REx, Rosetta, and Hayabusa 2. These were chosen due to the near-symmetric bodies, sizes (Bennu and Ryugu), and the complex geometry of comet 67P/Churyumov-Gerasimenko (potential for self shadowing due to lighting conditions).

Application to Bennu via PolyCam Images: Initial shape model generation was done using a collection of 222 images taken during the Approach Phase but only for the dates of November 2 and 3, 2018. During this time, the relative state was not as well known as later phases of the mission. Due to this, the range had the potential to not be as accurate. Regardless, the limb-based shape model routine was able to generate a relatively accurate shape model when compared to a Preliminary Sur-

vey SPC 75-cm shape model. Figure 7 shows a visual comparison with the SPC model.

Since Bennu was relatively small on the image plane, the limbs lacked the detailed required to generate a more rough, true surface. Additionally, the large protruding boulder, BenBen, did not have suitable viewing geometry to accurately model all of its sides. The difference in BenBen's shape can be seen in Figure 8. Bennu's view was rotated to show a different view of BenBen, bottom left side of the model image, where the largest difference is shown as dark blue in the histogram.

The model is not absent of differences. Table 1 details general numerical differences. As can be seen in the table and Figure 8, the vertex distance differences from a 75-cm SPC shape model of Bennu (from Preliminary Design) match reasonably well when estimating the center-of-mass of the object. Both the non-corrected and corrected centroid models have a mean difference that is larger than the original model. This could be due to a distance inaccuracy, limb estimation inaccuracy, or an unfitting Poisson Surface reconstruction smoothing depth, but there may be other causes as well. Generally, the vertex distance difference follows a near Gaussian distribution with the largest difference located at an unobservable

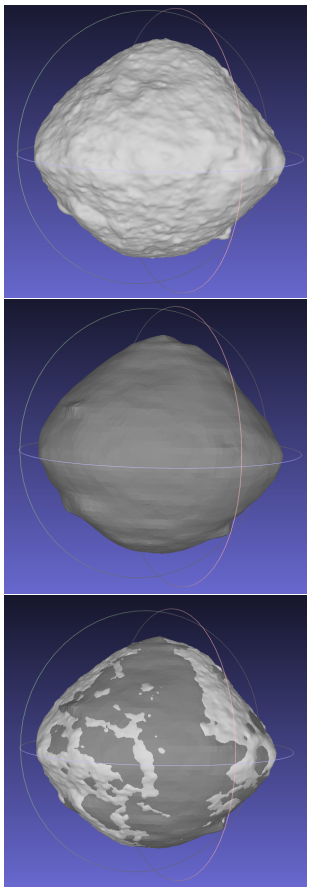
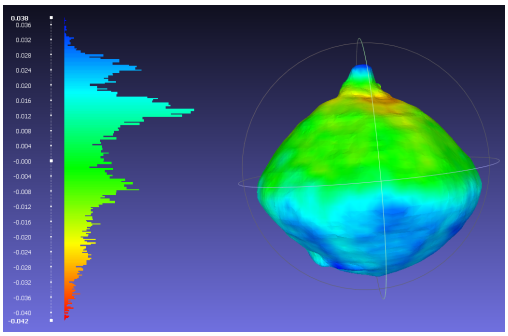
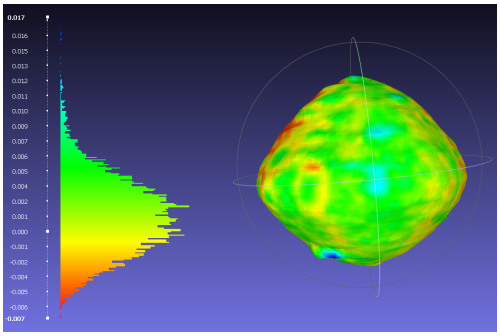


Figure 7: SPC 75-cm shape model (top) and Limb-based shape model with center-of-mass correction (middle) and both models overlay (bottom). Separate images are not aligned with each other



(a) No center-of-mass correction. Results: min difference -42.364m, max difference 37.99m, mean 3.368m, and RMS 16.949m

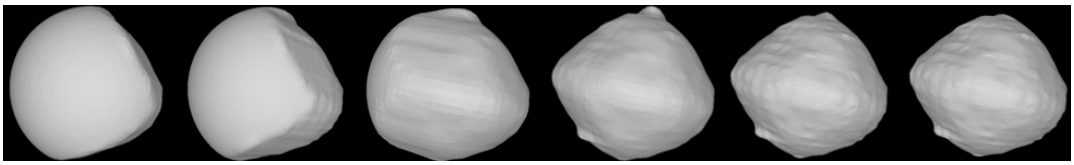


(b) With center-of-mass correction. Results: min difference -7.047m, max difference 17.347m, mean 1.122m, and RMS 3.219m

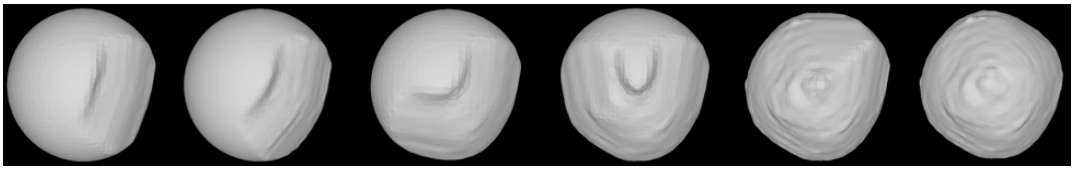
Figure 8: Limb-based shape model compared with SPC 75-cm shape model. Colorbars are in kilometers. Note that the colorbar ranges are different between (a) and (b)

Table 1: Limb-based shape model comparison where positive values indicate larger than the 75-cm SPC shape model

Parameter	Difference without $\bar{c}$ correction	Difference with $\bar{c}$ correction
Mean dif. [m]	3.368	1.122
RMS [m]	16.949	3.219
Min dif. [m]	-42.364	-7.047
Max dif. [m]	37.994	17.347



(a) View 1



(b) View 2

Figure 9: Shape model generation by using 110 images over 6 update steps. From left to right: +5 images, +5 images, +20 images, +20 images, +30 images, +30 images

location on the surface of the target. Different viewing angles would remedy this observability. However, due the the viewing geometry not changing much on approach, this seems unlikely to be rectified. Shape model updates can be done post approach to a target.

Model Update Process: Testing of the shape model update was done by using a collection of six updates. The first two updates only used a total of ten images. Thereafter, four more updates were done using the following image counts: +20 images, +20 images, +30 images, and +30 images, for a total of 110 images used. Figure 9 shows two views of the progress for the sequence of updates. Each update places more vertexes but trims others. As it can be seen, the sphere vertex trimming and filling is more apparent during the initial stages due to the lack of target surface coverage. For each update, the overall shape shape model is less than 1 MB, as is each vertex point file and its corresponding outward-pointing normal unit vector. However, the stored limbs file will increase in file size with the number of images added as well as increase of the number of limb points. Overall, the process works well and allows for a reduced upfront data requirement for shape model generation.

Raspberry Pi 1 Implementation: Development of the method herein took into consideration the computational cost of operating on a single board computer. The Raspberry Pi 1 seemed a suitable choice due the RAM and CPU limitations. For the Raspberry Pi 1 implementation, the CPU was throttled down to approximately 70 MHz, which is similar to the RAD750 (“240 million Dhrystone 2.1 instructions per second (MIPS) at 133 MHz”), a radiation-hardened single-board computer created by BAE Systems.[11] With this CPU speed reduction, the program was able to finish in about 15 minutes for processing 10 images and reconstructing the surface. Nine minutes of the runtime was spent reconstruction the surface, 1-2 minutes was spent loading in the images and SPICE files, and the remaining time was point generation and trimming (approx. 30 seconds per image). For a reference, an unthrottled CPU was able to complete the entire process in about one and a half minutes. Further work can be done to optimize the flight computer implementation.

Future work: For future development, a desire for more autonomy is in the forefront. There are a few avenues to explore. The following list are areas of consideration:

- Test irregular shape objects (example: comet 67 P/CG)
  - Incorporate any modifications to generalize tool
- Test algorithm on DART images
- Shape model use in proximity operations simulation
  - Compare to flight data and trajectories
- Range uncertainty to incorporate any scale errors
- Center-of-mass and orientation/pole estimation
  - Go beyond blob centroid and phase correction

- Terminator point ellipsoidal estimate constraint
- Investigate non-IAU standard rotation scenarios
- Implementation on an flight computer
- A true flight test

Conclusion: Work has been presented that enables initial shape model generation on approach to a target. A differing method to shape model generate allow for this capability to implemented on-board, giving an option of autonomy. This method does not require expensive downlink or shape model construction times. The method does rely on an understanding of the target's body fixed frame dynamics, camera pointing knowledge, and target's relative state to the spacecraft. However, the method demonstrated that a shape model may be generated on-board and that the shape model difference from a more accurate and detailed method (SPC) has small differences. Further work will be done to understand the possibility to incorporate estimation for these assumptions, allowing for further autonomy and less information downlinked to Earth as well.

References: [1] F. Preusker, et al. (2017) *Astronomy & Astrophysics* 607:L1. [2] N. Hirata, et al. (2019) *LPI Contributions* 2189:2093. [3] D. S. Lauretta, et al. (2021) in *Sample Return Missions* 163–194 Elsevier. [4] M. Nolan, et al. (2019) in *Lunar and Planetary Science Conference* 2132 2162. [5] M. Al Asad, et al. (2021) *The Planetary Science Journal* 2(2):82. [6] Y. Kuzu, et al. (1983) *Fourth Turkish-German Joint Geodetic Dags Vol PAMI-5* (2):150. [7] K.-m. G. Cheung, et al. (2005) *International Journal of Computer Vision* 63(3):225. [8] D. A. Baker, et al. (2019) in *2nd RPI Space Imaging Workshop*. [9] M. Kazhdan, et al. (2006) in *Proceedings of the fourth Eurographics symposium on Geometry processing* vol. 7. [10] F. Bernardini, et al. (1999) *IEEE Transactions on Visualization and Computer Graphics* 5(4):349 doi. [11] R. Berger, et al. (2001) in *2001 IEEE Aerospace Conference Proceedings (Cat. No.01TH8542)* vol. 5 2263–2272 vol.5 doi.