Probabilistic 3D Data Fusion for Adaptive Resolution Surface Generation

Andrew E. Johnson
Jet Propulsion Laboratory
California Institute of Technology
aej@jpl.nasa.gov

Roberto Manduchi
Computer Engineering Department
University of California, Santa Cruz
manduchi@seo.ucsc.edu

Abstract

In this paper we present an algorithm for adaptive resolution integration of 3D data collected from multiple distributed sensors. The input to the algorithm is a set of 3D surface points and associated sensor models. Using a probabilistic rule, a surface probability function is generated that represents the probability that a particular volume of space contains the surface. The surface probability function is represented using an octree data structure; regions of space with samples of large covariance are stored at a coarser level than regions of space containing samples with smaller covariance. The algorithm outputs an adaptive resolution surface generated by connecting points that lie on the ridge of surface probability with triangles scaled to match the local discretization of space given by the octree. To demonstrate the performance of our algorithm, we present results from 3D data generated by scanning lidar and structure from motion.

1. Introduction

To increase coverage and reduce spacecraft complexity, autonomous exploration by multiple complementary sensor platforms will be used for future solar system exploration missions. Examples are an orbiter, lander and surface rover used for small body exploration, or a fleet of aerobots exploring the surface of an outer moon. In these scenarios, data collected from multiple complementary vantage points is integrated into a single map. This map is shared between platforms and acts as an interface between platforms to enable formation guidance, navigation and control.

Using multiple platforms to map a surface has distinct advantages over single platform mapping. Multiple platforms allow for wider coverage of the surface being imaged. For example an orbiter can give a broad view of the terrain, an aerobot a higher resolution overhead view, and a rover can map localized high-resolution details of the surface. Also, different platforms can contain sensors with different modalities allowing for complementary multi-modal mapping of the surface. Finally, if multiple platforms are used, a broader coverage of the surface can be obtained in a shorter amount of time.

If it is to be used by all of the platforms, the sensor data from each platform should be incorporated into a single 3D map. 3D shape can be acquired directly from active optical sensors or indirectly through the use of passive sensors and machine vision techniques. Map building from multiple platforms has several unresolved issues including: how to combine data from different sensor modalities and different resolutions, each with different error characteristics; how to incorporate platform positional uncertainty into the map being built; and what map representation facilitates guidance and control.

Map building using a single mobile sensor has been studied extensively. In general, the techniques integrate geometric (2D or 3D) sensor data from multiple views into a single map that describes the shape of the surface being imaged. Various assumptions are used to make the problem tractable: 2 _ D representations, fixed data resolution, single object, or known sensor positions. These assumptions do not hold when mapping a surface using multiple distributed sensor platforms utilizing sensors of varying modality. The relative uncertainty in the position of the sensor platforms influences the map being built, so sensor data uncertainty as well as sensor platform uncertainty must be handled correctly. Also in some application scenarios the resolution of data from different sensors will vary dramatically (e.g., the case of an orbiter and a lander or rover), so multi-resolution modeling methods must be employed to seamlessly integrate data of varying resolution. Finally, the map will be built in parallel by multiple exploring platforms, so the map-building algorithm should address the serial/parallel nature of data acquisition.

Building a seamless surface from multiple overlapping 3D data sets has been studied extensively. Some of the first work focused on integrating data by creating an implicit function [1][14] and then polygonizing it using the marching cubes algorithm [8]. These algorithms were

implemented using a single resolution data structure. Hilton and Illingworth [6] have developed a multiresolution surface integration algorithm. This algorithm combines and compresses data using an octree, but it does not explicitly model sensor noise. Also, their algorithm does not generate adaptive resolution surfaces, although it is straightforward using their marching triangles algorithm [5].

Occupancy grids model the probability that a region of space is occupied by combining noisy sensor data using Bayesian statistics. Occupancy grids have been shown to be very effective for robot navigation, and 2D [3] and multiresolution 3D [11] versions of occupancy grids have been developed. The main drawback of occupancy grids for our application is that they do not generate a surface representation; they are inherently a volumetric data structure. However, the concepts employed to generate occupancy grids did provide us with a conceptual starting point for our algorithm development.

The input to our adaptive resolution surface generation algorithm is a set of 3D surface samples collected using different sensors of varying location and modality. Each sample is represented by a measurement model derived from a sensor model and estimates of registration error. All 3D sensors, devices that measure 3D shape directly, and 3D structure recovery algorithms have some form of measurement model that encodes the uncertainty in the 3D measurement. Using measurement models, data collected from different sensors at different locations and resolution can be combined in a formal way.

Using a probabilistic rule, a surface probability function (SPF) is generated that represents the probability that a particular volume of space contains the integrated surface. For efficiency, the SPF is represented using an octree data structure; regions of space with samples of large covariance are stored at a coarser scale than regions of space containing samples with smaller covariance. The SPF is built up incrementally from the samples; with each new sample, an increment to the SPF is computed based on the sample measurement model, and the data stored in the octree is updated.

The integrated surface lies along the ridge of surface probability. Finding the ridge requires second order differential computations. One approach to ridge finding would be to compute second order derivatives after the SPF is generated; however, this process is sensitive to discretization errors and can produce erroneous derivatives. Instead, the approach we take is to build the SPF *and* its derivatives incrementally from the 3D samples. During SPF generation, we analytically compute the changes to the SPF and its derivatives produced by each new sample and store them in the octree. After SPF generation ridge points are detected based solely on information stored in individual nodes of the octree, which makes ridge detection more accurate and robust.

The adaptive resolution integrated surface, represented as a triangle mesh, is generated by applying a variation of the marching triangles algorithm [5]. In our implementation of marching triangles, points on the integrated surface are found by connecting points on the ridge of surface probability. The size of the triangles generated is matched to the local discretization of space given by the octree, so regions of coarse surface sampling will have fewer faces than regions with fine surface sampling.

The details of the algorithm are explained in the following sections. To demonstrate the performance of our algorithm, we also present results that combine data generated using scanning lidar and structure from motion.

2. Ridge Detection in a Volume

Our algorithm for adaptive resolution surface generation first generates a 3D function defining the probability of surface in a volume and then detects the ridge of surface probability in this volume. Qualitatively, a point is on a ridge if the function achieves a maximum at that point along one or more directions in the space over which the function is defined. To be more rigorous, we use the height definition of a ridge stated by Eberly et al. [2] as follows.

Consider a function $f(x_1, x_2, x_3)$: ^{3} □ that is C^2 (partial derivatives to order 2 are continuous). Let f_{x_i} be the first partial derivative of f with respect to x_i. The gradient of f is the vector of first partial derivatives of f

$$\boldsymbol{g} = \left[f_{x_1}, f_{x_2}, f_{x_2} \right]^T.$$

$\boldsymbol{g}$ points in the direction of greatest change of f. The Hessian of f is the matrix of second order partial derivatives of f

$$\boldsymbol{H} = \begin{bmatrix} f_{x_1x_1} & f_{x_1x_2} & f_{x_1x_3} \\ f_{x_1x_2} & f_{x_2x_2} & f_{x_2x_3} \\ f_{x_1x_3} & f_{x_2x_3} & f_{x_3x_3} \end{bmatrix}$$

$\boldsymbol{H}$ defines the curvedness of f. f and $\boldsymbol{g}$ can be approximated in the region around $\boldsymbol{p}$ using

$$\begin{aligned} & f(\boldsymbol{x}) \approx f(\boldsymbol{p}) + (\boldsymbol{x} - \boldsymbol{p})^T \boldsymbol{g}(\boldsymbol{p}) + \tfrac{1}{2}(\boldsymbol{x} - \boldsymbol{p})^T \boldsymbol{H}(\boldsymbol{p})(\boldsymbol{x} - \boldsymbol{p}) \\ (1) \quad & \boldsymbol{g}(\boldsymbol{x}) \approx \boldsymbol{g}(\boldsymbol{p}) + \boldsymbol{H}(\boldsymbol{p})(\boldsymbol{x} - \boldsymbol{p}) \\ & \boldsymbol{H}(\boldsymbol{x}) \approx \boldsymbol{H}(\boldsymbol{p}) \end{aligned}$$

Let $\left[\lambda_1 \quad \lambda_2 \quad \lambda_3 \right]^T$ and $(\boldsymbol{v}_1^T, \boldsymbol{v}_2^T, \boldsymbol{v}_3^T)$ be the eigenvalues and eigenvectors of $\nabla\boldsymbol{H}$ where $\lambda_1 \geq \lambda_2 \geq \lambda_3$. $\boldsymbol{v}_1$ is the

direction in the volume where the gradient is changing the most. Positive (negative) values of κ_i correspond to concavities (convexities) of f in the direction $\boldsymbol{v}_i$. A point $\boldsymbol{x}$ is a ridge point of type 3-1 (i.e., a surface in a volume) if $\boldsymbol{v}_1(\boldsymbol{x})^T \boldsymbol{g}(\boldsymbol{x}) = 0$ and $\kappa_1(\boldsymbol{x}) > 0$. A point $\boldsymbol{x}$ is a strong surface ridge point if

(2) $\boldsymbol{v}_1(\boldsymbol{x})^T \boldsymbol{g}(\boldsymbol{x}) = 0$, $\kappa_1(\boldsymbol{x}) > 0$ and $\kappa_1(\boldsymbol{x}) > |\kappa_3(\boldsymbol{x})|$.

Essentially this definition says that a point is on a surface ridge when the component of the gradient in the direction of the greatest rate of change of the gradient is zero and the function is more concave than it is convex.

To determine if a point is on a ridge, the function, its gradient and its Hessian need to be defined at the point. In the next section, we show how we incrementally construct a surface probability function and its derivatives in order to facilitate ridge detection in a volume.

3. Surface Probability Function Derivation

The surface probability function (SPF) is a function that defines the probability that a region of space contains the surface. The SPF is generated by accumulating probabilities using the sensor models associated with multiple 3D measurements. The ridge in the SPF corresponds to a 2D boundary representation of the surface. Generating a SPF from multiple 3D measurements is straightforward, however, a way to generate the SPF that enables efficient and robust ridge detection is not immediately obvious. Below we describe how a SPF is generated from multiple 3D samples in a way that facilitates ridge detection.

3.1 Measurement Models

For the following discussion, we will consider 3D sensors, devices that measure 3D shape directly, and 3D structure recovery algorithms to be one in the same. Both generate measurements ζ in the form of 3D points $\boldsymbol{p}$ and both have associated 3D sensor models [1]. A *sensor model* is the probability density function used to determine the probability that the point $\boldsymbol{x}$ was actually the point measured by measurement ζ.

We define the *measurement model* to be the sensor model integrated over a small volume of space. The measurement model $M(\boldsymbol{x}) = P(\zeta \text{ measures } \boldsymbol{x})$ describes the probability that the region V of space around point $\boldsymbol{x}$ contains the surface. By ζ *measures* $\boldsymbol{x}$, we mean that the measurement ζ is generated by a surface element at $\boldsymbol{x}$. If V is small, then the sensor model will not vary significantly across V. In this case the measurement model is simply the product of the sensor model and V. Exactly how V is set will be explained in Section 4. Note that $M(\boldsymbol{x})$ is not a probability density function, it is the integral of the sensor model probability density function over a volume V. As $V \to \infty$, $M(\boldsymbol{x}) \to 1$ and as $V \to 0, M(\boldsymbol{x}) \to 0$.

In 3D structure recovery by stereo matching and triangulation, it has been shown [9] that the sensor model can be approximated by a gaussian G with mean centered on the measurement $\boldsymbol{p}$ and the covariance $\boldsymbol{\Sigma}_s$ dependent on stereo baseline, matched pixel coordinates and the covariance of the pixel tracking error. In the case of scanning lidar and radar, the measurement model is a function of the divergence of the beam, range accuracy of the sensor and properties of the surface being scanned. The lidar measurement model can also be approximated as a gaussian. Using gaussian approximations, both the stereo and lidar measurement model have the following functional form

(3) $$G(\boldsymbol{p}, \boldsymbol{\Sigma}_S) = \frac{e^{(\boldsymbol{x}-\boldsymbol{p})^T (\boldsymbol{\Sigma}_s)^{-1} (\boldsymbol{x}-\boldsymbol{p})}}{(8\pi^3 |\boldsymbol{\Sigma}_s|)^{1/2}}$$

The difference between the models comes in the definition of the covariance matrices $\boldsymbol{\Sigma}_s$. Recall that the measurement model is the product of the sensor model and a small volume V, around $\boldsymbol{x}$, over which the sensor model is relatively constant. Therefore, with a Gaussian sensor models, the measurement model is

$$M(\boldsymbol{x}) = G(\boldsymbol{p}, \boldsymbol{\Sigma}_S) \cdot V .$$

When the sensor is placed in a general pose described by a position $\boldsymbol{s}$ and a rotation matrix $\boldsymbol{R}$, the Gaussian measurement model is transformed to

$$M(\boldsymbol{x}) = G(\boldsymbol{R}\boldsymbol{p} + \boldsymbol{s}, \boldsymbol{R}^T \boldsymbol{\Sigma}_S \boldsymbol{R}) \cdot V$$

3.2 General Surface Probability Function

The SPF for a single measurement ζ, is generated from the measurement model as follows. The probability P that a point $\boldsymbol{x}$ is on the surface S of an object given the measurement ζ is

(4) $$P(\boldsymbol{x} \in S \mid \zeta) = P(\boldsymbol{x} \in S \mid \zeta \text{ measures } \boldsymbol{x}) \cdot P(\zeta \text{ measures } \boldsymbol{x}) + P(\boldsymbol{x} \in S \mid \overline{\zeta \text{ measures } \boldsymbol{x}}) \cdot P(\overline{\zeta \text{ measures } \boldsymbol{x}})$$

If all measurements come from the surface then the probability that $\boldsymbol{x}$ is on the surface given that the sensor actually measures $\boldsymbol{x}$ is 1. Furthermore, given an arbitrary point $\boldsymbol{x}$ and no measurements, it reasonable to assign _ to the probability that $\boldsymbol{x}$ is on the surface; we have no idea if the point is on the surface so give it a surface probability of 50%. Therefore, the probability that the point is on the

surface given that $\square$ does not actually measure $\boldsymbol{x}$ is still _ because in this case the sensor tells us nothing about $\boldsymbol{x}$. Also, $P(\overline{\square \text{ measures } \boldsymbol{x}}) = 1 \square P(\square \text{ measures } \boldsymbol{x})$, so (4) becomes

$$P(\boldsymbol{x} \square S \mid \square) = 1 \cdot P(\square \text{ measures } \boldsymbol{x}) + \frac{1}{2}(1 \square P(\square \text{ measures } \boldsymbol{x}))$$

$$P(\boldsymbol{x} \square S \mid \square) = \frac{1}{2}(1 + M(\boldsymbol{x}))$$

This probability is intuitive. If the $\boldsymbol{x}$ falls within the measurement model, then the surface probability will be greater than _ but less than 1; otherwise the probability is _.

The probability that a point $\boldsymbol{x}$ is on the surface given the measurements $(\square_1 \ldots \square_n)$ is $P^n(\boldsymbol{x}) = P(\boldsymbol{x} \square S \mid \square_1 \ldots \square_n)$. Using similar reasoning as above, $P^n(\boldsymbol{x})$ can be derived from multiple independent measurement models.

$$\begin{aligned} P^n(\boldsymbol{x}) &= P(\boldsymbol{x} \square S \mid \square_i \text{ measures } \boldsymbol{x}, \text{some } i) \cdot P(\square_i \text{ measures } \boldsymbol{x}, \text{some } i) + \\ &\quad P(\boldsymbol{x} \square S \mid \overline{\square_i \text{ measures } \boldsymbol{x}}, \text{all } i) \cdot P(\overline{\square_i \text{ measures } \boldsymbol{x}}, \text{all } i) \\ &= 1 \cdot (1 \square P(\overline{\square_i \text{ measures } \boldsymbol{x}}, \text{all } i)) + \frac{1}{2}(P(\overline{\square_i \text{ measures } \boldsymbol{x}}, \text{all } i) \\ &= 1 \square \frac{1}{2}(P(\overline{\square_i \text{ measures } \boldsymbol{x}}, \text{all } i)) \\ &= 1 \square \frac{1}{2}\prod_{i=1}^{n} (1 \square P(\square_i \text{ measures } \boldsymbol{x}) \end{aligned}$$

Substituting in $M^i(\boldsymbol{x}) = P(\square_i \text{ measures } \boldsymbol{x})$ and renaming the product to $f^n(\boldsymbol{x})$ yields the following definition for the SPF

$$P^n(\boldsymbol{x}) = 1 \square \frac{1}{2}\prod_{i=1}^{n} (1 \square M^i(\boldsymbol{x})) = 1 \square \frac{1}{2} f^n(\boldsymbol{x})$$

$$f^n(\boldsymbol{x}) = \prod_{i=1}^{n} (1 \square M^i(\boldsymbol{x}))$$

The SPF is generated incrementally by adding measurements one at a time to a volumetric data structure. Rather than storing $P^n(\boldsymbol{x})$, $f^n(\boldsymbol{x})$ is stored, because this is the only part of the SPF that changes. The incremental update rule for the $f^n(\boldsymbol{x})$ is

$$(5) \quad f^n(\boldsymbol{x}) = (1 \square M^n(\boldsymbol{x}))\prod_{i=1}^{n \square 1} (1 \square M^i(\boldsymbol{x})) = (1 \square M^n(\boldsymbol{x})) f^{n \square 1}(\boldsymbol{x})$$

The new product depends only on the old product and the measurement model for the current sample.

As described in the previous section, the first and second partial derivatives of the SPF are needed for ridge detection. The partial derivatives of SPF depend on the partial derivatives of $f^n(\boldsymbol{x})$; these can be represented with incremental update rules as well.

$$\begin{aligned} f^n(\boldsymbol{x})_i &= \square M^n(\boldsymbol{x})_i f^{n \square 1}(\boldsymbol{x}) + (1 \square M^n(\boldsymbol{x})) f^{n \square 1}(\boldsymbol{x})_i \\ (6) \quad f^n(\boldsymbol{x})_{ij} &= \square M^n(\boldsymbol{x})_{ij} f^{n \square 1}(\boldsymbol{x}) \square M^n(\boldsymbol{x})_i f^{n \square 1}(\boldsymbol{x})_j \square \\ &\quad M^n(\boldsymbol{x})_j f^{n \square 1}(\boldsymbol{x})_i + (1 \square M^n(\boldsymbol{x})) f^{n \square 1}(\boldsymbol{x})_{ij} \end{aligned}$$

These partial derivatives depend only on the previous partial derivatives, the measurement model and the measurement model derivatives for the current sample.

Once all of the measurements have been added $P^n(\boldsymbol{x})$ and its partial derivatives are computed from $f^n(\boldsymbol{x})$ and its partial derivatives using

$$(7) \quad P^n(\boldsymbol{x}) = 1 \square \tfrac{1}{2} f^n(\boldsymbol{x}) \quad P^n(\boldsymbol{x})_i = \square \tfrac{1}{2} f^n(\boldsymbol{x})_i \quad P^n(\boldsymbol{x})_{ij} = \square \tfrac{1}{2} f^n(\boldsymbol{x})_{ij}$$

3.3 Gaussian Surface Probability Function

The derivation above holds for sensors with arbitrary sensor models. All the results shown in this paper are for sensors with Gaussian sensor models. For this special case the partial derivatives of the Gaussian are required to generate the SPF and its derivatives using equations (5), (6) and (7). Let the Gaussian sensor model for the n^{th} measurement be $G^n(\boldsymbol{x}) = G(\boldsymbol{p}^n, \acute{\boldsymbol{O}}^n)$. Define $\boldsymbol{C}^n = \square \frac{1}{2}(\acute{\boldsymbol{O}}^n)^{\square 1}$ and let $\boldsymbol{C}_i^n$ be i th row of $\boldsymbol{C}^n$ and let $\boldsymbol{C}_{ij}^n$ be the element from the i th row and j th column of $\boldsymbol{C}^n$. The partial derivatives of the Gaussian measurement model $M^n(\boldsymbol{x}) = G^n(\boldsymbol{x}) \cdot V$ are

$$\begin{aligned} (8) \quad M^n(\boldsymbol{x})_i &= 2\boldsymbol{C}_i^n \cdot (\boldsymbol{x} \square \boldsymbol{p}) M^n(\boldsymbol{x}) \\ M^n(\boldsymbol{x})_{ij} &= (2\boldsymbol{C}_{ij}^n + 4(\boldsymbol{C}_i^n \cdot (\boldsymbol{x} \square \boldsymbol{p}))(\boldsymbol{C}_j^n \cdot (\boldsymbol{x} \square \boldsymbol{p})))M^n(\boldsymbol{x}) \end{aligned}$$

This completes the theoretical derivation of the surface probability function. In the next section we will describe how an SPF is generated using an octree data structure.

4. Octree Representation of the SPF

An octree is a volumetric data structure that partitions space in an efficient manner [13]. An octree starts with a root node that bounds the volume to be investigated for the given application. The root node can be partitioned into 8 children, each child corresponding to an octant of the volume spanned by its parent, the root node. The child nodes can be partitioned in a similar way down to the level where the leaf nodes are of the desired volume.

Usually an octree is used to store some data at a particular discretization of space. An octree is more efficient than a fixed resolution partitioning of space because only the nodes that contain data need to split down to the level used to store the data; regions of space that do not contain data can be represented at a coarser level.

4.1 SPF Storage in an Octree

In our application, we use an octree to store $f^n(\boldsymbol{x})$ and its derivatives. The node data structure we use contains the following members: (x,y,z) bounds on the volume spanned by the node, a pointer to the parent node, a list of pointers to the children nodes, and floating point data storage for the $f^n(\boldsymbol{x})$ (1 term), $f^n(\boldsymbol{x})_i$, (3 terms) and $f^n(\boldsymbol{x})_{ij}$ (6 terms since $f^n(\boldsymbol{x})_{ij} = f^n(\boldsymbol{x})_{ji}$). The (x,y,z) bounds of the node define a partitioning of space called a voxel. The center of the voxel is computed as the average of the bounds.

4.2 SPF Generation from Multiple Measurements

The SPF is built up incrementally from measurements and stored in an octree. In brief, the procedure for each measurement is: determine the octree level for inserting the measurement, determine the nodes at this level near the measurement, and update the SPF terms for each node. After all measurements have been added to the octree, the SPF and its derivatives are computed for each node using Equation (7). The details of the algorithm are described next.

First, the octree root node is initialized to a cube of side r that spans all of the data to be inserted into the volume. Usually all of the data is in available before integration, so the root is set to the bounding box of the data. If the data is not available, then the root bounds are set based the volume for which a surface representation is desired.

Each sample measurement $\boldsymbol{p}$ is inserted into the octree as follows. First, the desired sampling s of the measurement model is determined. In the case of a Gaussian measurement model the sampling is set to $|\Sigma|^{\frac{1}{6}}$ which is equivalent to one standard deviation for a spherical Gaussian. s determines the level l of the octree at which the measurement will be inserted by $l = log_2(r/s)$.

Next the number of samples m at sampling s that adequately spans the volume of the measurement model is determined. In the Gaussian case, m is set to 7, which creates a 3 standard deviation sampling of the volume around the measurement.

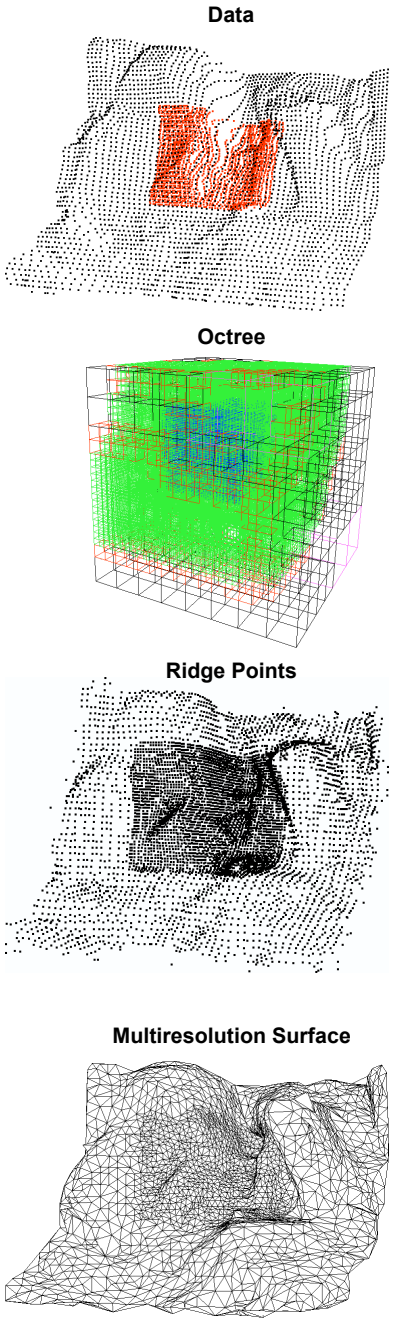

Figure 1 Steps in adaptive resolution surface generation for an example data set generated from two lidar scans of a pile of rocks. First the 3D measurements from the scans are inserted into and octree to generate the SPF. Next the ridge points are extracted from the SPF. Finally the ridge points are connected into a adaptive resolution surface mesh using marching triangles.

An $m \times m \times m$ volumetric grid of 3D sample positions $\boldsymbol{s}$ is then generated with spacing s centered on $\boldsymbol{p}$ (e.g., $(\boldsymbol{s} = \boldsymbol{p} + (is, js, ks)^T : -m \le i, j, k \le m)$). Next, for each $\boldsymbol{s}$, the node at level l that contains $\boldsymbol{s}$ is found by traversing the octree.

If the node is a leaf, the center $\boldsymbol{x}$ and volume V of the node are computed. $M(\boldsymbol{x})$ is then computed (at $\boldsymbol{x}$ using V) and the SPF terms $f^n(\boldsymbol{x})$, $f^n(\boldsymbol{x})_i$ and $f^n(\boldsymbol{x})_{ij}$ stored in the node are updated using equations (5) and (6).

If the node does not exist at level l, the smallest node containing $\boldsymbol{s}$ is found (the top node); the top node is then subdivided down to level l (the children nodes). The centers for the children nodes are computed. The initial SPF terms for children nodes are then determined by interpolating the SPF terms from the top node evaluated at the centers of the child nodes using Equation (1). The child node containing $\boldsymbol{s}$ is then updated as described for the leaf node above.

If the node is not a leaf, the tree is traversed to the leaves below the node, and the centers $\boldsymbol{x}$ of the leaves are determined. These leaf nodes are then updated as described for the leaf node above.

After all measurements have been added to the octree, in each node, $f^n(\boldsymbol{x})$ and its partial derivatives are replaced with the SPF and its partial derivatives computed using Equation (7).

Figure 1 shows an example surface integration problem. Two registered data sets, taken of a rock wall with a scanning lidar, are shown. The larger data set has 2x the spacing and 2x the variance in the measurement model when compared to the smaller data set. A rendering of the final octree generated to fuse the measurements is also shown; the region of the octree containing the smaller data set has nodes of a smaller size than the regions containing the larger data set. Also regions that do not contain data are not partitioned.

Figure 2 shows two slices of the SPF stored in the octree. One slice shows the SPF values stored in each node; the variable node size is clearly visible. The other slice shows the result of using quadratic interpolation of the SPF; in this case, SPF is much smoother. A smooth SPF is possible because the first and second order derivatives of the SPF are stored in each node. Given the smoothness of the SPF, sub-node ridge detection is possible.

The isoplot of the SPF shows that the surface probability is higher in the region associated with the smaller data set. This is to be expected because in this region the two data sets overlap so the number of samples per volume is higher and consequently the probability that a particular node contains the surface will be higher. If simple thresholding on surface probability was employed to detect the surface, then it is possible that only the surface corresponding to the small data set would be detected. In contrast, our approach to surface generation is insensitive to absolute magnitudes of surface probability because ridge detection is a second order process.

5. Ridge Extraction from Octree

In this section we describe how the nodes and resulting 3D points on the surface ridge are extracted from the octree storing the SPF.

Each leaf node in the octree contains the SPF and its first and second order derivatives which, according to (1), describe the local quadratic approximation to the SPF across the node. Using this information alone it is possible to determine if a ridge passes though the node and the position of points on the ridge. Consider a node with center $\boldsymbol{p}$. First the gradient $\boldsymbol{g}$ and Hessian $\boldsymbol{H}$ are constructed from the partial derivatives $P^n(\boldsymbol{p})$. Next the eigenvectors $(\boldsymbol{v}_1^T, \boldsymbol{v}_2^T, \boldsymbol{v}_3^T)$ and eigenvalues $[\lambda_1 \quad \lambda_2 \quad \lambda_3]^T$ for $-\boldsymbol{H}$ are computed where $\lambda_1 \ge \lambda_2 \ge \lambda_3$. According to (2), the conditions $\lambda_1 > 0$ and $\lambda_1 > |\lambda_3|$ are required for the node to contain a ridge. If these conditions are not met then the node is not a ridge node.

The next condition for a ridge is that $\boldsymbol{v}_1(\boldsymbol{x}) \cdot \boldsymbol{g}(\boldsymbol{x}) = 0$; if a node is to be considered a ridge node then $\boldsymbol{x}$ must be within the voxel corresponding to the node. Using (1) to approximate the $\boldsymbol{g}$, this condition becomes

$$(9) \quad \boldsymbol{v}_1(\boldsymbol{x}) \cdot \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{v}_1(\boldsymbol{p}) \cdot (\boldsymbol{g}(\boldsymbol{p}) + \boldsymbol{H}(\boldsymbol{p})(\boldsymbol{x} - \boldsymbol{p})) = 0$$

$\boldsymbol{v}_1(\boldsymbol{p})$ is an eigenvector of $\boldsymbol{H}(\boldsymbol{p})$, so $\boldsymbol{v}_1(\boldsymbol{p})^T \boldsymbol{H}(\boldsymbol{p}) = \boldsymbol{v}_1(\boldsymbol{p}) k_1$. This fact and some algebraic manipulations reduces (9) to

$$(10) \quad \boldsymbol{v}_1(\boldsymbol{p}) \cdot \boldsymbol{x} - (\boldsymbol{v}_1(\boldsymbol{p}) \cdot \boldsymbol{g}(\boldsymbol{p}) / k_1 + \boldsymbol{v}_1(\boldsymbol{p}) \cdot \boldsymbol{p}) = 0 .$$

This equation defines a plane corresponding to the ridge in the node. This makes intuitive sense because the plane, like the ridge, is a 2D surface in a 3D space. Any point on the plane can be selected as the ridge point as long as it lies within the voxel. However, for simplicity, we compute the point $\boldsymbol{x}$ on the plane closest to the center of the voxel $\boldsymbol{p}$ using

$$(11) \quad \boldsymbol{x} = \boldsymbol{p} + \boldsymbol{v}_1(\boldsymbol{p})(\boldsymbol{v}_1(\boldsymbol{p}) \cdot \boldsymbol{g}(\boldsymbol{p}) / k_1)$$

If $\boldsymbol{x}$ is within the volume bounds of the node and $\lambda_1 > 0$ and $\lambda_1 > |\lambda_3|$ then the node is a ridge node with ridge point $\boldsymbol{x}$ defined by (11).

After the SPF is generated, the values stored in each leaf node are replaced with (as will be described in the next section) the ones required for surface generation: P, $\boldsymbol{g}$, $\boldsymbol{v}_1$ and $[\square_1 \quad \square_2 \quad \square_3]$.

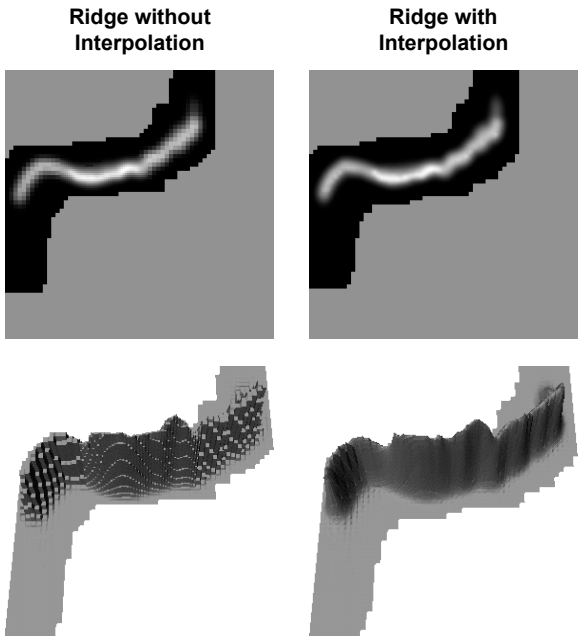

Figure 2 Two 2D slices through an SPF stored in an octree shown as grayscale images and isoplots. On the left is a slice showing the SPF values stored in each voxel that clearly show the discretization of the octree. On the right is a slice where the pixel values are interpolated across each voxel resulting in smooth transitions between voxels of different resolution and more accurate ridge detection.

Almost all of the ridge points lie on a single surface while a few of the ridge points are outliers. These outliers are caused by volume discretization and non-uniform spacing of measurements. Fortunately, most of the outliers can be eliminated using two simple checks. The first check removes ridge points with low surface probability while the second removes ridge points whose surface probability is less than the surface probability of the either of two neighboring nodes along the direction $\boldsymbol{v}_1$ (not a local maximum). Figure 1 shows the ridge points (after the checks stated above have been applied) detected for the data sets also shown in Figure 1.

6. Adaptive Resolution Surface Extraction

Many single resolution volumetric integration algorithms [1][14] build an implicit function sampled on a regular grid and then polygonize this function using the marching cubes algorithm [8]. Applying marching cubes to data stored at multiple resolutions is not straightforward because the data are no longer stored in a regular grid making the table lookup for polygonization difficult or impossible. Marching cubes can be applied to regions that have nodes of the same size, but when nodes of different sizes are adjacent, cracks will appear in the surface. This cracking problem can be solved by triangulation and hole filling[10][11] but these methods are undesirable because they generate faces that are perpendicular to the surface.

Marching triangles [5] is an efficient alternative to the marching cubes algorithm because it typically generates fewer faces. Marching triangles also applies volumetric Delaunay constraints to generate triangles that are as compact (close to equilateral) as possible. Furthermore, the size of the triangles generated can be controlled to fit the variable sampling present in octrees. The added benefits of marching triangles come at the cost of a more complicated algorithm.

Briefly, our implementation of marching triangles proceeds as follows. First find three points that are close to each other and are on the surface. From these points, initialize the surface mesh by generating three vertices, one triangular face and three edges. Select an edge from the triangle. Define a point by projecting out from the edge a fixed distance d perpendicular to the edge and in the plane of the triangle. Find the closest point on the surface to this point. If the surface point passes the Delaunay constraints (no other points are in the sphere made by the point and the face edge) then add the point and the resulting triangle to the face. If the point does not pass the Delaunay constraints then try and add the triangle made from the edge and one of the vertices neighboring the edge in the mesh. If these triangles do not pass the Delaunay constraint, then search for a point in the mesh that is across from the edge, but still within the projection distance d, and make a triangle from the edge and this point. If a point cannot be found or the generated face does not pass the Delaunay constraint then move on to the next edge.

We have modified the marching triangles algorithm to generate adaptive resolution surfaces by connecting ridge points stored in our octree data structure. The main changes come in the way that the first triangle is initialized, how the projection distance is defined and how the closest point on the surface is determined.

The projection distance is set to the length of the diagonal of the voxel that contains the vertices of the current edge. If the vertices lie in voxels of different size, the projection distance is set to the average of the diagonals. By setting projection distance based on voxel size, the size of the triangles generated will vary linearly with sampling of the SPF by the octree. Consequently, the resolution of the surface is adapted to match the spacing of the measurements used to generate the surface.

Finding the closest point on the surface is straightforward. Suppose you have query point $\boldsymbol{q}$ contained in a leaf node with center $\boldsymbol{p}$, gradient $\boldsymbol{g}$, principal curvature direction $\boldsymbol{v}_1$, and size s. If the leaf node is a ridge node then surface has been found and the ridge point for the node is returned. If not then the next leaf node, that is closer to the surface ridge, is found by traversing the octree to find the node that contains $\boldsymbol{q} = \boldsymbol{p} + \boldsymbol{v}_1(\boldsymbol{g} \cdot \boldsymbol{v}_1 / \|\boldsymbol{g}\|)s$. The term $\boldsymbol{g} \cdot \boldsymbol{v}_1 / \|\boldsymbol{g}\|$ is used to make sure that $\boldsymbol{v}_1$ points in a direction that is toward the surface ridge. This process is repeated until a ridge node is found or a node with no data is encountered. If no ridge node is found, the search terminates and failure is reported.

The first triangle is initialized at the maximum of surface probability. First the octree is traversed to find the ridge node with the maximum probability. The ridge point for this node is used as the first vertex of the triangle. The other vertices of the triangle are generated by finding ridge points that are in the nodes neighboring the maximum ridge node.

Figure 1 shows the surface generated for the data also shown in Figure 1. Note that surface is seamless, and contains two regions: one high resolution and one low resolution.

7. Results

Figure 3 shows the adaptive resolution surface resulting from the integration of two lidar scans of a target wall. The scans were generated during a test of hazard detection algorithms for safe landing on Mars [7]. During this test, a scanning lidar was placed on a rocket sled along with other instrumentation and onboard computing. The rocket sled was propelled down a track toward a 12x65m target wall composed of conex containers and acrylic hemispheres of varying diameter. As the target moved down the track multiple scans of the target were taken with the lidar. After the test, the motion of the sensor during scanning was removed from the measurements using a sensor trajectory (position and attitude) computed using from onboard sensors.

The lidar has a divergence of 0.1° and a range accuracy of 1% of the range. The fine scan was taken at 100 m, which results in beam width of 17 cm and a range accuracy of 10 cm. The coarse scan was taken at 200 m, which results in beam width of 35 cm and a range accuracy of 20 cm. For simplicity, the measurement models of the two scans were approximated by spherical Gaussians with standard deviations of 15 cm and 30 cm. The measurement models were large enough to incorporate the sensor measurement errors as well as the error in trajectory generation used to compensate the measurements for the motion of the sled. Given the extent of the data and the size of the measurement models the fine scan was inserted at level 7 in the octree while the coarse scan was inserted at level 6.

The resulting adaptive resolution surface has the expected characteristics. The surface is more resolved in the regions that have the fine data and the two regions are seamless integrated across their border. A comparison of the reconstructed surface to a CAD model generated for the target wall shows a good fit; large-scale features with sufficient measurements are reconstructed correctly while features with few measurements are smoothed to the surface supported by the majority of the measurements.

Our lab has a planetary imaging testbed to collect imaging and lidar measurements in a controlled setting. The testbed consists of an XY gantry that moves a prismatic joint in the Z-axis. At the end of the Z-axis is a camera on a pan/tilt unit and a single axis scanning lidar. The sensors are placed above a box containing rocks and sand used to simulate the appearance of planetary terrain.

Figure 4 shows the adaptive resolution surface generated from measurements collected using the planetary imaging testbed. A coarse data sets is generated by taking multiple single axis scans with the lidar as it is moved along the horizontal X axis of the gantry. The angular accuracy of the scanner is 6 mm and the range accuracy is 10 mm. To account for these errors and discontinuities in the data, the measurement model is set to a spherical Gaussian with a standard deviation of 20mm. The coarse measurements are insert at level 6 in the octree.

The fine data set is generated by moving the camera across the scene to collect a sequence of images. These images are processed to estimate the motion of the camera and then dense structure from motion is use to reconstruct the imaged scene. The scale of the scene is set based on the translational motion between images measured by encoders on the gantry. The measurement model for the fine data set is set to a spherical gaussian with a 6 mm standard deviation based on the camera model and the baseline between the images used for structure recovery. The fine measurements are inserted at level 8.

The two data sets are coarsely aligned manually and then accurately aligned using the iterative closest point algorithm.

The resulting adaptive resolution surface shows the seamless integration of the data sets that differ in resolution by a factor of 4 (two levels in the octree). This result also shows that our algorithm can fuse measurements from different sensing modalities taken from widely different positions and resolution.

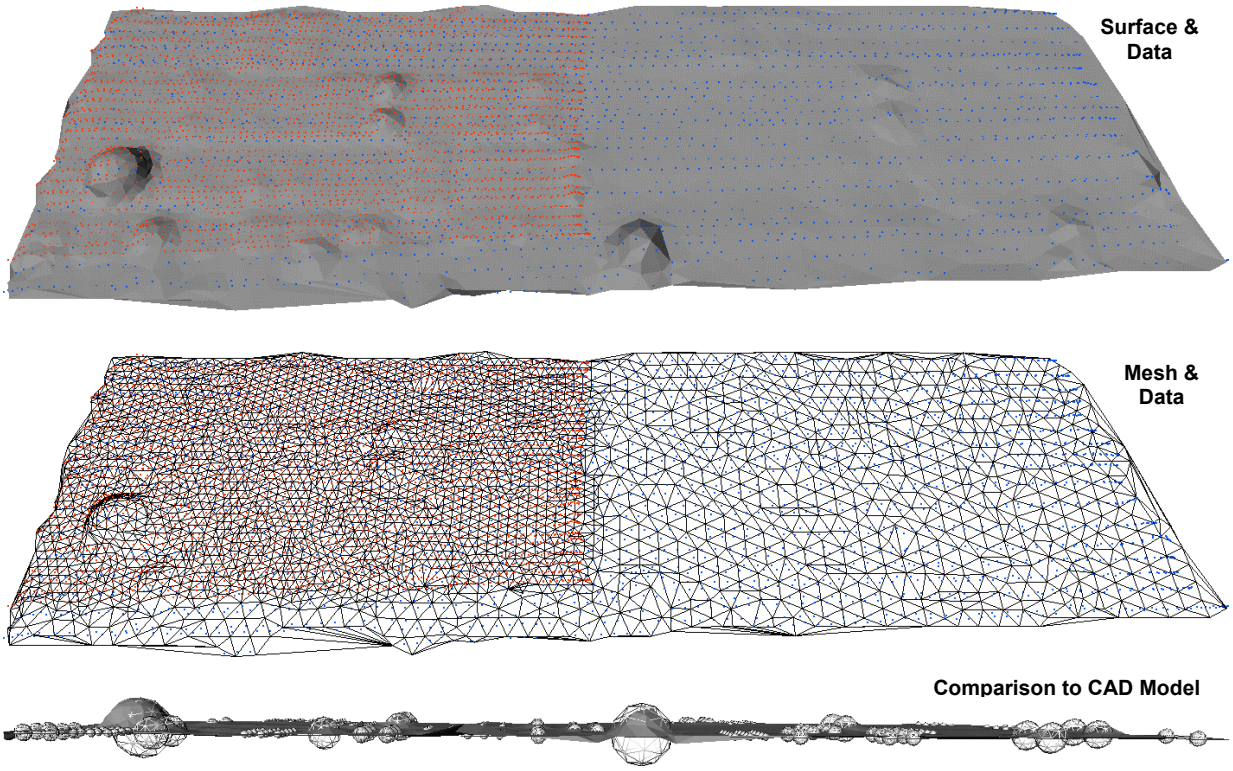

Figure 3 Adaptive resolution surface generated from two scans of a target wall used in a test program for safe landing on Mars. The coarse scan was taken ~200 m from the wall and the fine scan was taken ~100 m from the wall. Comparison to a CAD model for the target wall clearly shows that large scale features with sufficient measurements are reconstructed correctly while features with few measurements are smoothed to the surface supported by the majority of the measurements.

8. Conclusions

To reduce spacecraft complexity and meet mission goals with limited resources, future planetary missions will replace monolithic spacecraft equipped with multiple sensors by multiple distributed spacecraft each with a single sensor. To enable concurrent usage by all of the platforms, the sensor data from each platform will be incorporated into a single 3D map. To this end, we have developed an algorithm for adaptive resolution surface generation that combines data taken with multiple types of sensors, at multiple resolutions and from arbitrary positions. Our surface generation algorithm deals effectively with noisy data in a simple and formal probabilistic manner.

This work is the starting point for multiple areas of future work. Since the SPF is represented in an octree, it can be compressed using standard octree compression techniques. For example when the 8 children of a node all contain similar data, they can be reduced to a single node. Consequently, during adaptive resolution surface generation, larger triangles will be generated in and around this node. Another area of research will be the compression and transmission of the integrated surface. The surface can be represented at many levels: samples and measurement models, the surface probability function, or the adaptive resolution surface. Each representation has its own form of compression. Choosing the correct data structure for transmission will depend on the application, computing resources and bandwidth available to the multiple distributed assets employed in mapping.

References

[1] B. Curless and M. Levoy, “A Volumetric Method for Building Complex Models from Range Images,” SIGGRAPH `96, 1996.

[2] D. Eberly, R. Gardner, B. Morse, S. Pizer and C. Scharlach, “Ridges for Image Analysis,” Jour. Mathematical Imaging and Vision 4(4):353-373, 1994.

[3] A. Elfes, "Using Occupancy Grids for Mobile Robot Perception and Navigation," Computer Magazine, pp. 46-57, June 1989.

[4] G. Guy and G. Medioni, “Inference of Surfaces, 3D Curves, and Junctions from Sparse, Noisy 3D Data,” IEEE Trans. Pattern Analysis and Machine Vision 19(11):1265-77, November 1997.

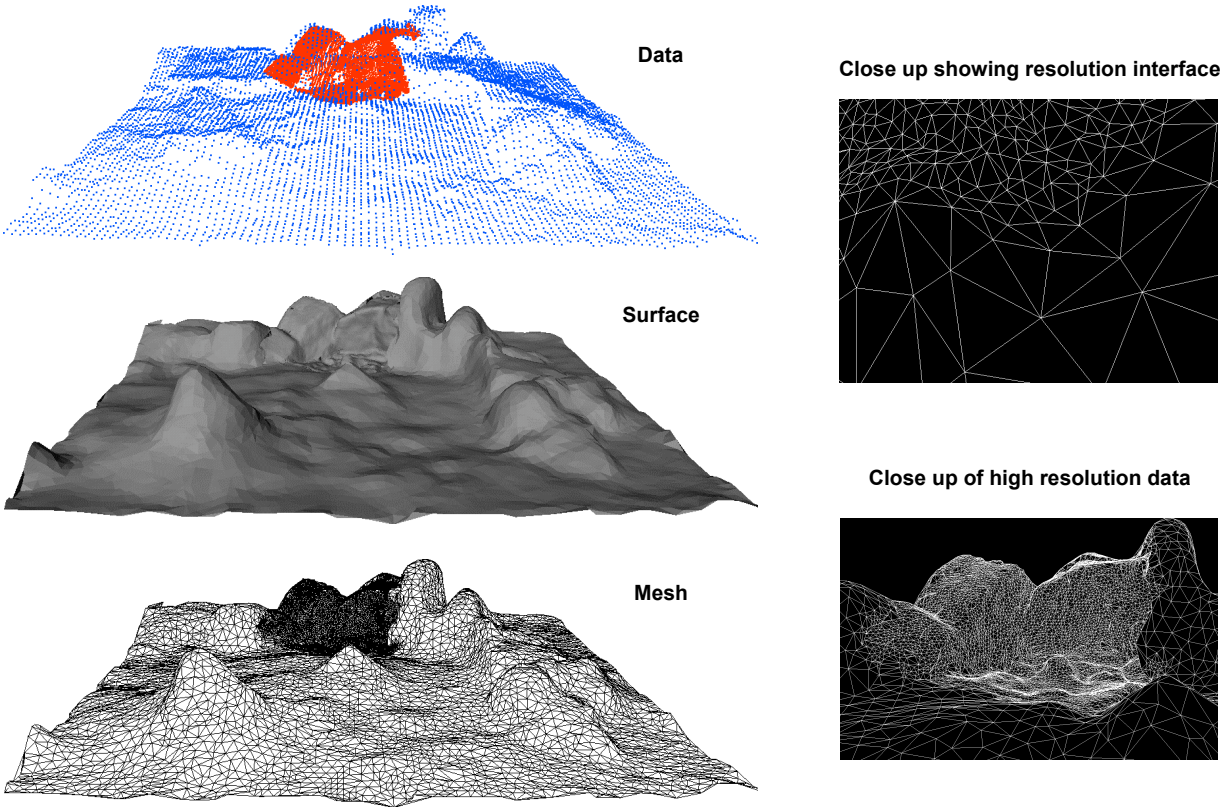

Figure 4 Adaptive resolution surface generated from measurements collected with a planetary imaging testbed using two different sensing modalities. The coarse data set was generated using overhead single axis scanning lidar (10 mm spacing). The fine data set was collected using a single camera and structure from motion 3D reconstruction (2 mm spacing). A close up of the overlapping regions shows the smooth integration of the data sets with more detailed features being present in the region containing the fine data. A close up of the mesh border between the data sets shows the seamless transition between triangles of small size to triangles of a large size.

[5] A. Hilton, A. Stoddart, J. Illingworth and T. Windeatt, "Marching Triangles: Range Image Fusion for Complex Object Modeling," Proc. Int'l Conf. Image Processing (ICIP96), 1996.

[6] A. Hilton and J. Illingworth, "Multi-Resolution Geometric Fusion," Proc. Int'l Symp. Recent Advances in 3D Digital Imaging and Modeling (3DIM97), pp. 181-188, 1997.

[7] A.E. Johnson and E. D. Skulsky, "Descent Speed Testing of a Hazard Detection System for Safe Landing on Mars," *Proc. 25th AAS Guidance and Control Conference 2002*, AAS-02-024, 2002.

[8] W. Lorensen and H. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," Computer Graphics, 21(3): 163-9, 1987.

[9] L. Matthies and S. Shafer, "Error Modeling in Stereo Navigation," IEEE Jour. Robotics Automation 3(3): 239-48, 1987.

[10] G. Nielson, D. Holliday and T. Roxborough, "A Solution to the Cracking Problem Based upon Triangular Coons Patches," Proc. IEEE Visualization 1999 (VIS99), 1999.

[11] P. Payeur, P. Herbert, D. Laurendeau, C. Gosselin, "Probabilistic Octree Modeling of a 3-D Dynamic Environment," Proc. IEEE Int'l Conf. Robotics and Automation, pp. 1289-1296, 1997.

[12] R. Shekhar, E. Fayyad, R. Yagel and J.F. Cornhill, "Octree-Based Decimation of Marching Cubes Surfaces," Proc. Visualization 1996 (VIS96), pp. 335-42, 1996.

[13] R. Szeliski and S. Lavallée, "Matching 3-D Anatomical Surfaces with Non-Rigid Deformations using Octree Splines," IJCV 18(2): 171-186, 1996.

[14] M. Wheeler, Y. Sato, and K. Ikeuchi, "Consensus surfaces for modeling 3D objects from multiple range images," Proc. Int'l Conf. Computer Visions (ICCV '98), pp. 917 – 924, 1998,.

Acknowledgements

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. We would like to thank Chuck Bergh for building the planetary imaging testbed and David Skulsky for managing the safe landing task that provided the target wall data.