

# A Human Supervisory Approach to Rapid World Modeling Through the Use of Geometric Primitives<sup>1</sup>

**Jason Luck**

**ESA-EPE MS J580**

**Los Alamos National Laboratory**

**Los Alamos, NM 87545**

**Randy Roberts**

**ESA-EPE MS J580**

**Los Alamos National Laboratory**

**Los Alamos, NM 87545**

## Abstract

A three-dimensional world model is crucial for many robot-oriented tasks. The most efficient mapping configuration uses geometric primitives to model environments. These primitive shapes use very little memory for large world maps, such as industrial environments, and are easy to store and process. In the past, modeling techniques have been either fully manual or autonomous. Manual methods are extremely time consuming but also highly accurate and flexible. On the other hand autonomous techniques are fast but inflexible and often inaccurate. The method presented in this paper combines the two thereby yielding a highly efficient, flexible, and accurate tool. Our methods enable a human supervisor to quickly construct a fully defined world model from unfiltered and unsegmented real-world range data.

**MASTER**

## 1.0 INTRODUCTION & BACKGROUND

### 1.1 Introduction

**DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED**

There are many uses for intelligent robots in industrial environments. In most cases, accurate maps of these unstructured environments are not available. For these situations, the ability to construct an accurate world map is a necessity for robotic deployment [1]. The data used in this project consists of real-world data taken using a structured lighting system at Sandia National Laboratories. The data is very noisy and contains errors that render most segmentation algorithms inadequate. A high concentration of reflectance points (artifacts of the structured lighting system) permeate the data set and cause large variations in surface normal and curvature values for all points surrounding the reflectance point [2]. Other errors include additive noise, which also causes surface normal and curvature deviations. In addition, the data points are non-uniformly sampled, meaning the distance between two neighboring points on the x-sampling axis can be much larger than the distance between two neighboring points on the y-sampling axis. This non-uniformity causes preferential growth in the direction of higher point density and failure of many region-growing algorithms. Registration errors cause large abnormalities when multiple images of the same scene are overlaid. Moreover, in the data appear large obscured

1. This work was partially funded by the DOE Robotic Technical Development program and partially funded by the Defense Advanced Research Projects Agency under ARPA order number AO 8029 Amendment #6.

### **DISCLAIMER**

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

# **DISCLAIMER**

**Portions of this document may be illegible  
in electronic image products. Images are  
produced from the best available original  
document.**

regions, which cause undefined intersections between planes and holes in the map. As a result of these inaccuracies in the data, conventional modeling methods produce inaccurate and incomplete world maps. This paper describes some new robust methods that overcome the complexity associated with such real-world data.

## 1.2 Background

The traditional method of generating a world map is to manually create models using CAD tools [3]. This method is not only extremely time consuming but may be impossible in hazardous areas in which object measurements cannot be taken. Fully autonomous methods have also been tested in recent years, but these methods do not work reliably enough with real-world data on a task in which an inaccurate model could lead to a hazardous spill [4]. In this case, the best method is to create a mapping routine that is as autonomous as possible but with a human supervisor to guide the process. Many modeling techniques use free-form surface models. These approaches approximate the surface using techniques such as B-splines, NURBS, or Dual Kriging [5,6]. These techniques create a smoothed surface map, but the image must be segmented before the process begins. Noise and occlusion also have a large effect on the surface models, thereby making conversion to a solid model error prone. Transformation of a surface mesh to a solid model has been accomplished [7], but the process requires multiple scans of the scene so that all edges and all boundaries are identified. To deal with occlusion and noise, as well as to make object recognition and grasping easy, primitive models that approximate the shape and size of an object are required. Many papers approach the subject of creating a world map based on geometric object models (see [8,9,10]). Several of these approaches can find the position and orientation of an object, but they need a predetermined model. Others adeptly create models from segmented data but do not deal with obscured regions, object intersections, or approach the problem of segmentation. Many methods use edge detection to create well-defined object intersections and to aid segmentation [11,12,13]. These methods work well on complete data sets but have problems when edges are obscured. Because of the problems associated with data of this type, the approach presented in this report was developed.

## 2.0 APPROACH

Our environment and modeling paradigm is as follows: Let  $S_i = \{(x, y, z)\}$  be the  $i^{th}$  of set of range samples collected from surfaces in an environment. The samples  $S_i$  are typically collected from a ranging system (e.g., structured lighting) and provide a single view of the environment. Consider the union of  $N$  sets of range samples of the same scene but collected from different viewpoints. The set of range samples,  $S = S_1 \cup S_2 \dots \cup S_N$ , thus represent samples of surfaces in the environment. Our objective is to build a reliable model from the data in  $S$ . In particular, we focus on industrial environments. Our environment model consists of three types of surfaces: bounded planes, bounded cylinders, and bounded random surfaces. These shapes were selected because in an industrial environment, such as a hazardous waste site, most objects can be represented

by planes and cylinders. Objects that cannot be defined by these two primitives are modeled to a random surface using a type of Voronoi tessellation. Our model  $M$  can thus be expressed as

$$M = \bigcup_{i=1}^L P_i \bigcup_{j=1}^M C_j \bigcup_{k=1}^N R_k$$

where  $P_i$  represents the  $L$  planes in the scene,  $C_j$  represents the  $M$  cylinders in the scene, and  $R_k$  represents the  $N$  random surfaces in the scene.

Because of the large extent of errors in real-world data and the lack of reliability in autonomous methods, we have opted to use a human supervisor to guide the modeling process. The supervisor selects seed points on the surfaces of objects and instructs the computer to segment and model them. From a processing standpoint, the objects are first segmented from the data in  $S$  and then modeled appropriately. Segmentation is performed by rotating the data in  $S$  to favorable coordinate systems, followed by segmentation. The segmented data is then fit to a model. Only the x-y-z position of the data points is used in segmenting and modeling. Other measures are avoided, allowing the data to be processed without a filtering or smoothing stage. Once objects are modeled, they are extrapolated and interpolated until a fully defined world model is produced.

Our work employs two different options for choosing the seed points that start the modeling process: either a single point or a small set of points can be used. The single point method is a step towards autonomy, but several additional routines are required for autonomy. A routine would be needed to generate appropriate seed points, like the method implemented by Besl [14]. Another routine would involve determining the type of surface the seed point is on; curvature may be sufficient to make this classification. A last routine would have to determine if the fitted geometry is an acceptable fit; such routines were used by Leonardis [15] and Bolles[16]. The current routines allow for either seed point strategy, although a human supervisor is still needed to direct the segmenting. We begin by describing the segmentation and characterization of bounded planes.

### 3.0 DETAILED DESCRIPTION

#### 3.1 Planes

The supervisor typically starts by building planes in a scene. A bounded plane is described by two sets of parameters: a vector  $p$  that parameterizes the unbounded (infinite extent) plane and a vector  $e$  of edge points. Vector  $p$  contains four coefficients  $[a, b, c, d]$  of the equation of a plane,  $ax + by + cz = d$ , whereas the vector  $e$  consists of triplets of points  $(x, y, z)$  that define the edges (boundaries) of the plane. An estimate of

the vector  $p$  is computed from user-defined seed point triplets,  $u_1 \dots u_m$ , using a least-squares procedure, as shown.

$$X = (F^\dagger F)^{-1} F^\dagger G, \text{ where}$$

$$F = \begin{bmatrix} u_{1x} & u_{1y} & 1 \\ u_{2x} & u_{2y} & 1 \\ \vdots & \vdots & \vdots \\ u_{mx} & u_{my} & 1 \end{bmatrix}, G = \begin{bmatrix} u_{1z} \\ u_{2z} \\ \vdots \\ u_{mz} \end{bmatrix}, X = \begin{bmatrix} -a & -b & d \end{bmatrix}, c = 1,$$

$^\dagger$  is the transpose operator, and the vector  $p$  is contained in  $X$  and  $c$ . The best results are obtained when seed points are evenly spaced around the plane. From this a rotated coordinate system (primed system) is constructed such that the plane described by  $p$  lies in the  $(x', y', 0)$  plane [17]. The rotation matrix  $R$  allows transformation between the world coordinate system and the primed system. The entire data set  $S$  is rotated into the primed coordinate system. Data in  $S'$  whose  $z'$  coordinate is not within a factor of the standard deviation  $\sigma_z$  of the initialization plane are discarded. Effectively, a one-dimensional (i.e., in the direction perpendicular to the plane) segmentation of the data is performed, thereby effectively interpolating shadows and holes in the range imagery into the plane. Denote the new set of segmented data as  $T'$ , where  $T' \subseteq S'$ . All that remains is to perform a two-dimensional segmentation of the data in  $T'$ .

The samples in  $T'$  belong to the infinite plane formed from the initial seed points. To complete the description of the bounded plane, the samples in  $T'$  are segmented in the  $(x', y')$  directions to find the bounding points,  $e$ . Segmentation in the  $(x', y')$  is performed as follows. First of all, the seed points,  $u_i$ , form a bounding box. The box width and height then expand to twice the original size and eliminate all points exterior to the bounding box. Next, all points are projected onto the  $x'$  axis; consequently, the region along the axis where points occur is split into ten smaller regions of equal length. The smaller regions are inspected, and if one contains less than 15% of the maximum number of points in any region, the routine discards the points in the region and all points from that region to the current plane boundary. The same process then occurs on the  $y'$  axis. If points are discarded, the entire process is repeated until no new points are eliminated. Next, a function inspects the outermost regions for any *wing points* that may be present. Wing points appear when searching for plane points on the side of a box. Points on the intersection line along the floor will also satisfy the plane criteria and form wings on the side of the box, as shown in figure 1.

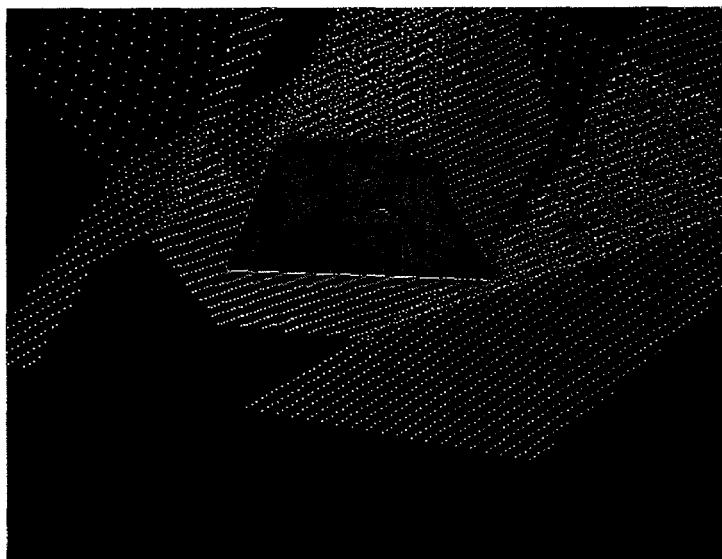


FIGURE 1. Three overlaid scans of a box resting on the floor.

Accordingly, wing points are positioned linearly along either the top or the bottom of the plane. The function adjusts the extent of the plane in the positive and negative  $x'$  directions to values where points both above and below the average  $y'$  value exist. The process is then repeated along  $y'$  axis, thereby removing any wing points. Finally, the convex hull of the remaining samples in  $T'$  is computed. The convex hull forms the edges,  $e'$ , of the region in the primed coordinate system. The plane parameters and edges of the plane in the world coordinate system are found by rotating  $p'$  and  $e'$  back into the world coordinate system.

A single point can also be used to find a plane within the scene. The same process is used to determine the plane parameters, except that the single seed point,  $u_1$ , cannot determine the equation of the infinite plane. Therefore additional seed points,  $u_2 \dots u_n$ , must be found to feed into the least-squares algorithm. This is accomplished in two steps: (1) finding  $u_2 \dots u_m$  and (2)  $u_m \dots u_n$ . The routine finds the 50 points closest to the seed point,  $u_2 \dots u_n$ , and performs an initial estimation of the plane equation using a least-squares fit. As before, a rotation matrix,  $R$ , rotates the points into primed system. Next, the distance to the seed point and the elevation angle off the plane are calculated for each point. The points furthest from the seed point and closest to the average elevation angle in each direction are saved,  $u_m \dots u_n$ . The seed point,  $u_1$ , its closest fifty neighbors,  $u_2 \dots u_m$ , and the newly found points,  $u_m \dots u_n$ , are used to estimate the plane through the least-squares fit procedure. The routine, as before, develops a rotation matrix,  $R$ , and eliminates all points in  $S'$  whose  $z'$  coordinates are not within a factor of the standard deviation,  $\sigma_z$ , of the initialization plane. Segmentation occurs exactly as in the previous method, except that the initial bounding box is not calculated from the seed points

because only one seed point was selected. The one-point method works reliably, but it is important to choose the seed point so that the closest 50 points are all within the plane.

### 3.2 Cylinders

A bounded cylinder is defined by four parameters: a scalar radius,  $r$ ; an axis vector,  $\mathbf{a}$ ; a center point to fix the axis vector,  $\mathbf{c} = (x_c, y_c, z_c)$ ; and a matrix that represents the boundary (height) of the cylinder,  $\mathbf{h}$ . Modeling is accomplished using the Levenberg-Marquardt least-squares procedure, which requires initial estimates of the parameters. Processing begins by estimating the axis vector of the cylinder [18]. Four seed points,  $\mathbf{u}_i = (x_i, y_i, z_i)$ , chosen with the first two pointing approximately along the axis, are used to form a region on the side of the cylinder. The seed points are designated by green + signs and points within the defined region are shown in red (figure 2). The region is divided into halves, and two parallel vectors are sought with one in each half. The vectors are shown in yellow, extending from points a1 to b1 and from points a2 to b2. Note that

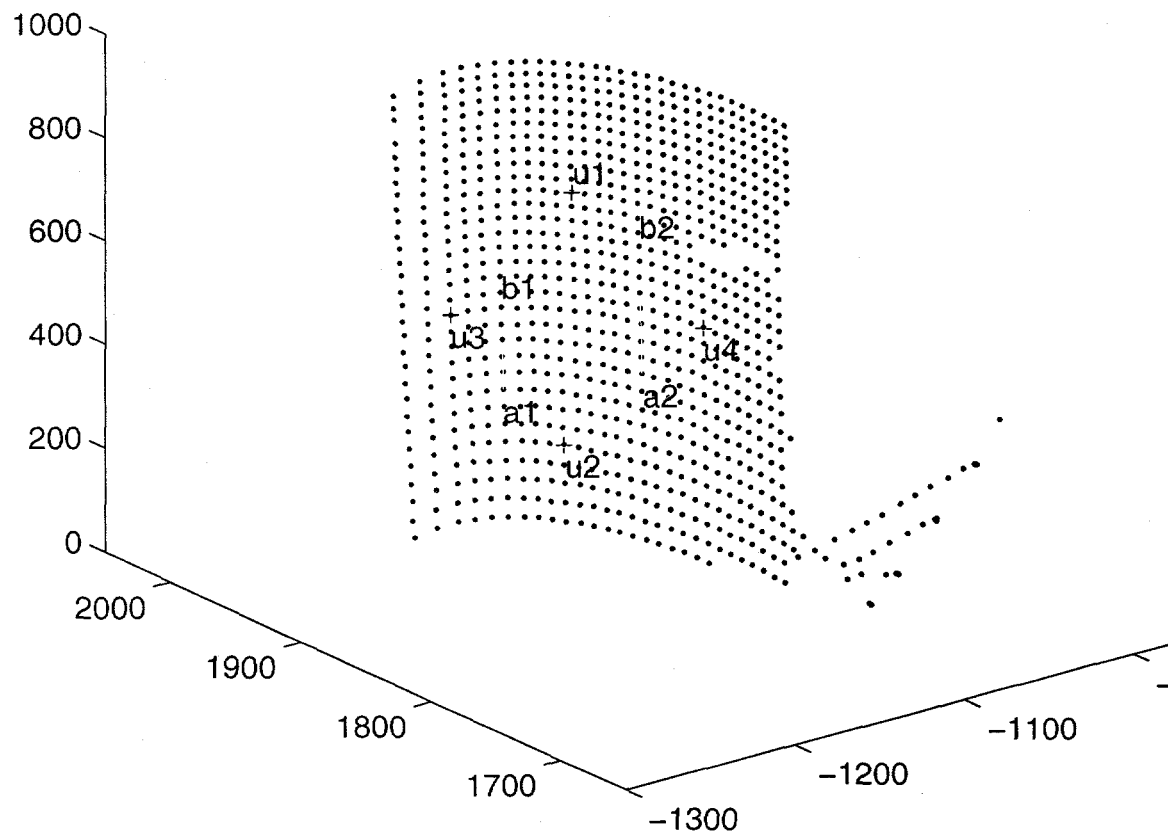


FIGURE 2. A partial cylinder undergoing axis estimation.

if two parallel vectors are found on the surface of a cylinder, they must also be parallel to the cylinder axis. The estimate of the cylinder's axis,  $\hat{\mathbf{a}}$ , is used to construct a rotation matrix,  $\mathbf{R}$ . Data in  $S$  is then rotated into the primed coordinate system so that the cylinder axis is parallel to the  $z'$  axis. Projecting data in  $S'$  onto the  $(x', y')$  plane results in an arc



of samples for an incomplete cylinder, or an entire circle for a complete cylinder (see figure 3). Now that the data has been projected onto the  $(x', y')$  plane, parameters  $r$  and  $c$  can be estimated. Estimation of  $r$  and  $c$  consists of two steps: first, a heuristic but robust technique is used to segment cylinder samples from other data in  $S$  and provide initial estimates; and second all the parameters are refined with a least-squares algorithm.

Recall from elementary geometry that the center of a circle can be found at the intersection of two perpendicular bisectors of any two chords of the circle. The radius of the circle is then found as the distance from the center of the circle to any point on the circle. The transformed seed points  $u_i'$  are known to lie within the arc of points formed by projecting the cylinder onto the  $(x', y')$  plane. These points are connected into chords in the circle and used to estimate the center of the circle  $\hat{c}$ . An initial estimate of the radius,  $\hat{r}$ , is found as  $\hat{r} = |\hat{c} - u_i'|$ . Estimates  $\hat{c}$  and  $\hat{r}$  can be improved by using larger chords,

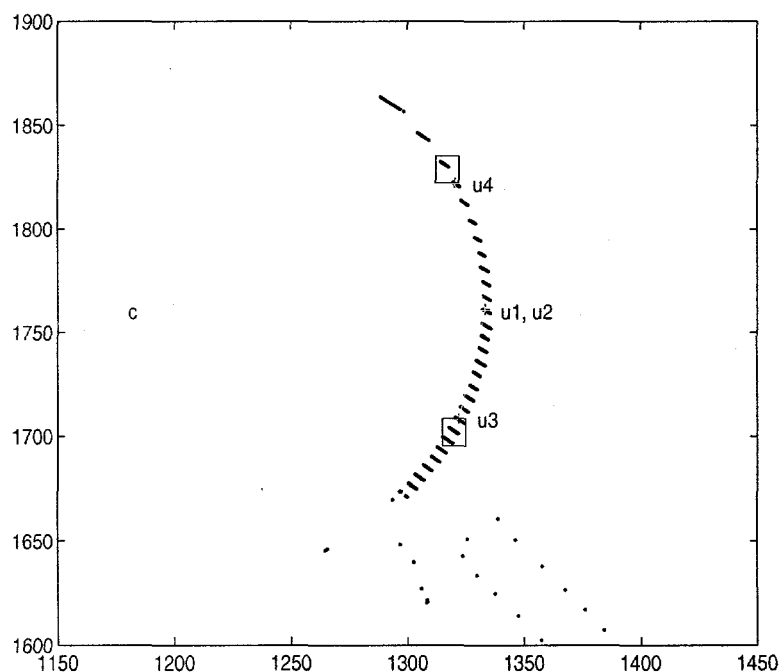


FIGURE 3. The cylinder has been rotated so the main axis parallel to the  $z'$  axis.

which can be found by iteratively moving the ends of the chords,  $u_3'$  and  $u_4'$ , to the ends of the arc. Small neighborhoods (shown in red) surrounding  $u_3'$  and  $u_4'$  are first defined by the seed point itself, a point using the same radius but 7.5 degrees closer to the end of the arch, and by a factor times the deviation of the radius of all points within the previous neighborhood  $\sigma_r$  (for the first iteration  $\sigma_r$  is set to 1). If points are found within the neighborhood bounds, either  $u_3'$  or  $u_4'$  is redefined at the center of the neighborhood

bounds and  $\hat{c}$  and  $\hat{r}$  are re-estimated. Continued iterations provide refined estimates of  $\hat{c}$  and  $\hat{r}$ , and also an estimate of the variance of the radius estimate  $\sigma_r$ . The variance estimate, together with  $\hat{c}$  and  $\hat{r}$ , allow samples from the cylinder to be segmented from the samples in  $S'$ . These segmented samples are called  $T'$ , where  $T' \subseteq S'$ .

The final estimates of cylinder axis, center, and radius are provided by a least-squares procedure operating on data in  $T'$ , using the previously obtained estimates as initialization values. Although seven parameters are needed to define a cylinder, adjustment of only five of these components will obtain a fit. To find a center point,  $c$ , on the main axis, the  $z$  component remains constant while the  $x$  and  $y$  components are adjusted until they lay on the main axis line. To find a vector parallel to the main axis line,  $a$ , the  $z$  component again remains fixed while the  $x$  and  $y$  components change until the vectors are parallel. The Levenberg-Marquardt least-squares method adjusts these four parameters along with the radius to obtain a fit for the following model. In a cylinder, the radius,  $\hat{r}$ , is the distance between a data point and a point on the main axis where the vector from the data point to the axis point is perpendicular to the main axis. This statement defines two equations that yields the difference between the estimated parameters and each data point: Equation 1

$$r^2 - (D - (c + t \times a))^2 = \sigma \quad (1),$$

where  $D$  = Data point  $(x,y,z)$ , and  $\sigma$  = the deviation of each point from the cylinder model (ideally 0), and Equation 2 which is drawn from the statement defining the parameter  $t$ :

$$[a] \cdot [D - (c + t \times a)] = 0 \quad (2).$$

The process minimizes the deviation of each data point by adjusting the model parameters using the partial derivatives of each parameter. The process iterates itself until it converges upon the best set of parameters.

Several different values of cylinder height are possible. For example, the height of a right cylinder can be described by a single number, whereas a cylinder with arbitrary extent can require a complex description of its height. Anticipating cylinders with complex ends, we describe the cylinder's height as a set of coordinates  $h$ . Data that forms the cylinder is rotated and translated so that the  $z'$  is parallel to the cylinder axis, and the origin of the primed coordinate system is located at the center of the cylinder. The cylinder is segmented into 10-degree sectors, and the maximum and minimum  $z'$  value is found for each sector. These values form the set  $h'$ , which when rotated back into the world coordinate system describe the extent of the cylinder.

A single seed point can also begin the cylinder parameter estimation process. In this case, the routine follows the same process with some modifications to the axis estimation and segmentation routines. Because a single seed point cannot form a bounding box, a different set of criteria must determine which points are to be used in the search for parallel vectors. A simple routine slowly increases the extents of a bounding box around the seed point in each direction until a point is absorbed in each direction. Using these distances, the box extents increase by the designated amount in each direction until 100 points are absorbed. Because noise can cause very short vectors to be parallel even if they are not parallel to the main axis, the calculation eliminates points close to the seed point. The remaining points are scanned for two parallel vectors extending in opposite directions from the seed point.

The segmentation algorithm also required modifications to adjust to the lack of seed points. In the four-seed point method, the seed points are used in both the initial approximations and as starting points of the arc-sweeping process. The two points of the 100 found earlier that are farthest from the seed point and on opposite sides of the cylinder are replace the missing seed points. These points are then fed into the same algorithms, with parameter estimation and segmentation followed by the least-squares fit and height estimation. Again, the one-point method works reliably when the seed point is chosen so that the neighboring 100 points are all on the cylinder.

### 3.3 Random Surfaces

Random surfaces are modeled as a connected group of triangles. First, the points are rotated and projected onto a plane. And second, a two-dimensional Voronoi tessellation creates a connectivity list for a section of the points chosen by the user. Therefore, both the points and a rotation must be specified by the user. Two methods are provided to accomplish this. The user can rotate the points until in the desired viewport and drag a square around the desired points. In this case, the chosen points are rotated so that the  $z'$  axis points straight out of the screen. The user can also define a plane and expand a bounding box until all points desired for the tessellation are within the box, as shown in step 2 of example 2. In this case, the data will be rotated so that the defined plane is the  $(x', y')$  plane. A standard Voronoi tessellation routine [19] then creates a connectivity list, ignoring the  $z'$  component. With the  $z'$  component re-incorporated, the data is rotated back to the world coordinate system, where triangles are drawn between the points according to the connectivity list.

## 4.0 MODEL BUILDING & EDGE EXTRAPOLATION

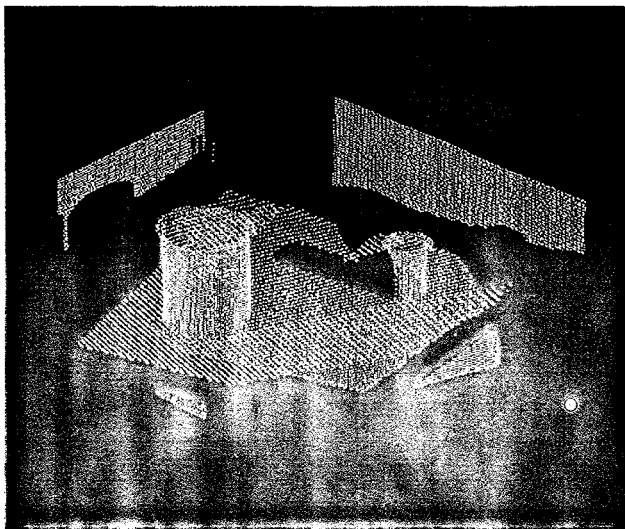
Most applications involving robots require a CAD model to guide the robot. Therefore, transformation from a group of objects to a fully defined world model is essential for robotic deployment. Because the objects we have modeled are saved as equations of solid models, transformation to a world CAD model is trivial. Before this can be accomplished, obscured regions must be interpolated and objects must be extrapolated so that well-defined intersections occur.

Our method allows object intersections to be calculated from the parameters of the models. The intersection of two non-parallel planes is a line. The routine calculates the equation of this line and then computes plane points along the line so that the planes meet on the line without crossing over it. A new convex hull for each plane is then calculated,  $e$ , with these new points included and with any points extending past the intersection eliminated. The new plane convex hulls will have a well-defined intersection with a known equation. This method can be extended to encompass three planes, such as the corner of a room. An automated routine uses multiple calls of the two-plane intersection routine to transform the planes so that a well-defined corner appears. If a cylinder is standing on a plane, it too can be modified so that the intersection is smooth and the cylinder lies on the plane. Because the equation of both the cylinder and the plane are known, the height points,  $h$ , around the outside of the cylinder can be either extended or drawn back until they lay in the plane. When the cylinder is redrawn, it lays flat on the plane and the intersection is known. The process of intersecting objects interpolates obscured regions and creates a world model with well-defined edges.

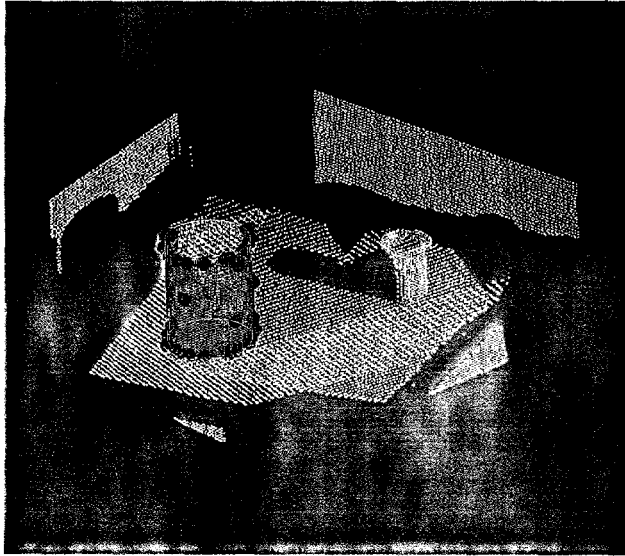
The same idea also applies to boxes. As a result of occlusion, boxes occasionally appear as only three planes. The other sides of the box are not always visible to the sensors, but three sides are sufficient to accurately reconstruct the box. The assumption that a box side is parallel to and the same size as the opposing side allows the routine to reproduce all six sides of a box from three adjacent planes. First, the routine calculates the equation and convex hull of each of the three adjacent planes. The routine then forms a coordinate system with the intersection of the three planes as the origin and each axis pointing along a plane intersection line. The dimensions of the box are equal to the maximum value in each axis direction. Using the previously stated assumptions, the routine creates points at the corners of the box and draws planes connecting the corner points.

## 5.0 EXAMPLES (TWO EXAMPLES ARE PROVIDED TO ILLUSTRATE THE MODELING TECHNIQUE)

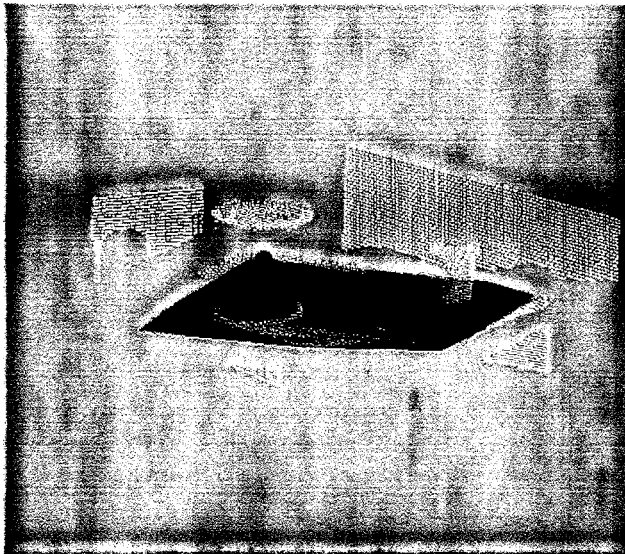
### 5.1 Example 1 (2 cylinders in a corner taken from 2 overlaid scans)



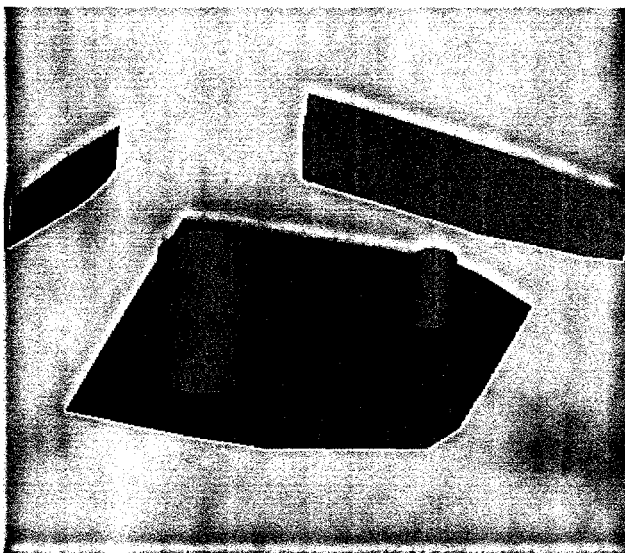
Step 1: The original data set is shown. It contains two scans of a scene with two cylinders in the corner of a room.



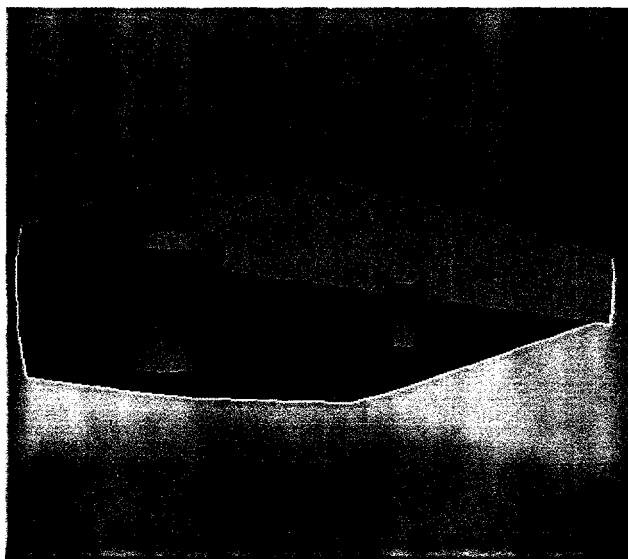
Step 2: The wire-frame of a modeled cylinder has been over-laid on the data. The large green points are the seed points. The smaller highlighted green points are the points included in the least-squares fit. The purple points are the height points.



Step 3: To model a plane seed points (green) have been chosen around the plane. The modeled bounded plane is shown in purple and the highlighted red points are the points included within the plane extents.

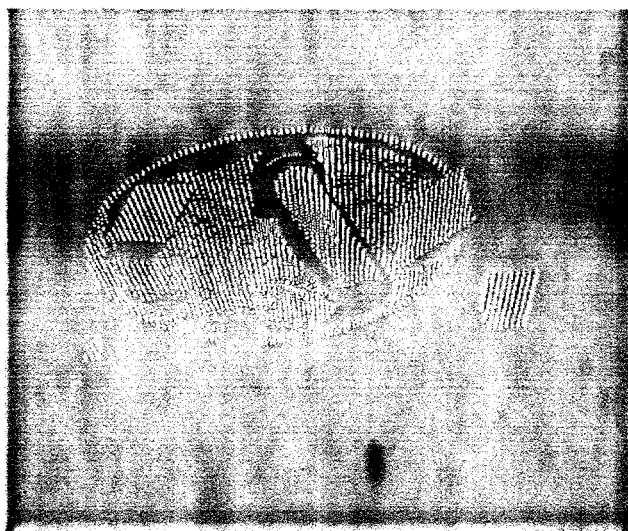


Step 4: All the planes and cylinders have been modeled and the models are saved in a separate viewer.

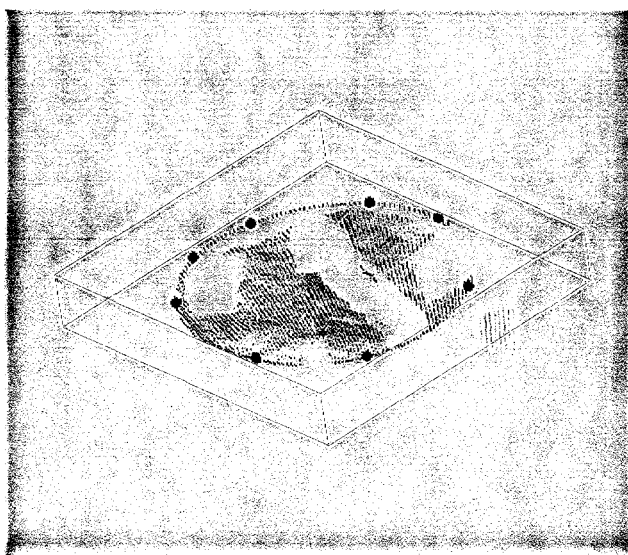


Step 5: The planes have been interpolated with one another so that the extents are well defined. The cylinders have also been extrapolated into the floor so they lay flat on the floor.

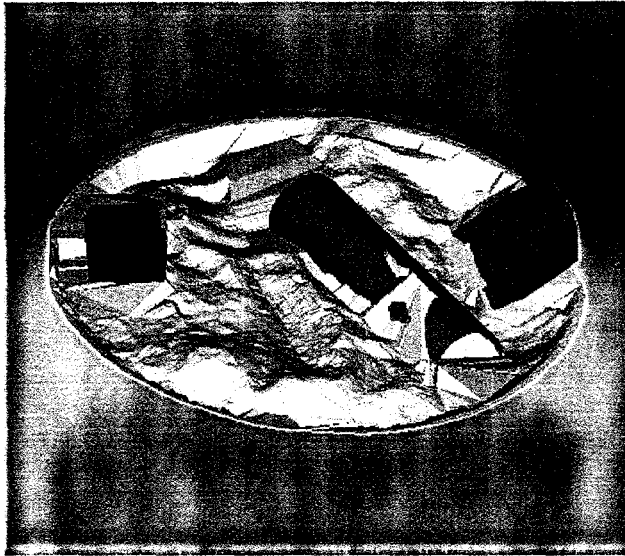
## 5.2 Example 2 (2 boards and a cylinder in a vermiculite pit)



Step 1: The original data set is shown. It contains two planes (boards) and a cylinder in a vermiculite pit.



Step 2: The cylinder and the two planes have been modeled. Seed points are chosen to obtain a viewport-defining plane and the bounding box is expanded to include all points desired for the Voronoi tessellation (blue).



Step 3: The finished model is shown with the planes and cylinder incorporated in the voronoi tessellation of the vermiculite pit.

## 6.0 IMPLEMENTATION

Our system uses a combination of AVS5 visual software and MATLAB mathematical software. AVS5 is used to build a graphical interface and to view the data and world map. MATLAB was used to write the routines for segmenting and modeling the data. In general, AVS worked well as a graphical interface. Rotations, translations, and point and object picking are easily accomplished. In addition, building the interface for controlling operations was fairly simple. MATLAB worked extremely well in addressing the large data sets and its high-level programming features are compatible with the tasks associated with this project. The MATLAB code had to be translated in C language so that AVS could read the code. The MATLAB C compiler was used for this purpose and was easy to use. The compiler had errors in a few functions: `sort()`, `min()`, and `max()`, but routines were written to avoid the problems. MATLAB now has a new version of the compiler that fixes these problems.

The only problem with the combination was in data transfer. MATLAB uses all double precision variables and the routines were written using three-dimensional arrays. AVS did not use the same setup; therefore, before MATLAB functions were called, the arrays would have to be transformed into MATLAB arrays and then the returning arrays would have to be transformed back into the AVS format. This process is time consuming and memory intensive, but if this problem is overlooked the combination works extremely well.

## 7.0 FUTURE WORK

- Address the problems incurred when transferring data between MATLAB routines and AVS routines.
- Test the code on scenes where a ground truth is known.
- Develop code to intersect objects with random surfaces.

- Determine the usefulness of cones and spheres as additional object models (depending on the application).
- Determine the feasibility of creating a fully autonomous system using the one-point algorithms.

## 8.0 ACKNOWLEDGMENTS

The authors would like to acknowledge Janine Fales, ESA-DE, Los Alamos National Laboratory, and Steve McBurnett, DARPA, for supporting this effort. We would also like to thank Charles Little, Chris Wilson, and Colin Selleck at Sandia National Laboratories for their help.

## 9.0 BIBLIOGRAPHY

- [1] R. E. Barry, C. Q. Little, and B. L. Burks, "Requirements and Design Concept for a Facility Mapping System," Proc. ANS 6th Topical Meeting on Robotics and Remote Systems, Monterey CA, pp.775-783, Feb. 5-10, 1995.
- [2] R. Hoffman and A. K. Jain, "Segmentation and Classification of Range Images," IEEE Transactions of Pattern Analysis and Machine Intelligence, vol. pami-9, no. 5, Sept. 1987.
- [3] P. J. Flynn and A. K. Jain, "CAD-Based Computer Vision: From CAD Models to Relational Graphs," Proc. 1989 IEEE Int. Conf. on Systems, Man and Cybernetics, pp. 162-167, Cambridge, MA, 1989.
- [4] A. Hoover, G. Jean-Baptiste, X. Jiang, P. Flynn, H. Bunke, D. Goldgof, K. Bowyer, D. Eggert, A. Fitzgibbon, and R. Fisher, "An Experimental Comparison of Range Image Segmentation Algorithms," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 18, no. 7, pp. 673-688, July 1996.
- [5] I. D. Faux and M. J. Pratt, *Computational Geometry for Design and Manufacture*, Ellis Horwood Limited, England, 1985.
- [6] A. Limaïem and H. A. ElMaraphy, "Automatic Inference of Parametric Equations in Geometric Modeling Using Dual Kriging," Proc. 1996 IEEE Int. Conf. on Robotics and Automation, Minneapolis, MN, pp. 1499-1504, 1996.
- [7] M. K. Reed and P. K. Allen, "A Robotic System for 3-D Model Acquisition from Multiple Range Images," Proc. 1997 IEEE Int. Conf. on Robotics and Automation, Albuquerque, NM, 1997.
- [8] W. A. Hoff, F. W. Hood, and R. H. King, "An Interactive System for Creating Object Models from Range Data based on Simulated Annealing," Proc. 1997 IEEE Int. Conf. on Robotics and Automation, Albuquerque, NM, 1997.
- [9] J. Feddema, and C. Little, "Rapid World Modeling: Fitting Range Data to Geometric Primitives," Proc. 1997 IEEE Int. Conf. on Robotics and Automation, Albuquerque, NM, 1997.
- [10] A. Johnson, P. Leger, R. Hoffman, M. Hebert, and J. Osborn, "3-D Object Modeling and Recognition for Telerobotic Manipulation," IEEE/RSJ International Conference on Intelligent Robots and Systems, Pittsburgh, PA, Aug. 5-9, 1995.
- [11] J. K. Hackett and M. Shah, "Segmentation Using Intensity and Range Data," Optical Engineering, vol. 28, no. 28, June 1989.



- [12] M. A. Wani and B. G. Batchelor, "Edge-Region-Based Segmentation of Range Images," *IEEE Transactions of Pattern Analysis and Machine Intelligence*, vol. 16, no. 3, March 1994.
- [13] K. Umeda, K. Ikushima, and T. Arai, "3-D Shape Recognition by Distributed Sensing of Range Images and Intensity Images," *Proc. 1997 IEEE Int. Conf. on Robotics and Automation*, Albuquerque, NM, 1997.
- [14] P. J. Besl and R. C. Jain, "Segmentation Through Variable-Order Surface Fitting," *IEEE Transactions of Pattern Analysis and Machine Intelligence*, vol. 10, no. 2, March 1988.
- [15] A. Leonardis, A. Gupta, and R. Bajcsy, "Segmentation of Range Images as the Search for Geometric Parametric Models," *International Journal of Computer Vision* **14**, 253-277 (1995).
- [16] R. Bolles and M. Fischler, "A RANSAC-Based Approach to Model Fitting and Its Application to Finding Cylinders in Range Data," *Proceedings 7th International Joint Conference on Artificial Intelligence*, Vancouver BC, Canada, Aug. 24-28, 1981.
- [17] R. Roberts, "An Algorithm for Segmenting Range Imagery," Los Alamos National Laboratory Report LA-UR 96-4497, November 1996.
- [18] W. E. L. Grimson, T. Lozano-Perez, and D. P. Huttenlocher, *Object Recognition by Computer: The Role of Geometric Constraints*, MIT Press, Cambridge Mass., 1990.
- [19] S. J. Fortune, "A Sweepline Algorithm for Voronoi Diagrams", *Algorithmica* **2**, 153-174 (1987).