MANUFACTURED TO AIM STANDARDS
BY APPLIED IMAGE, INC.

Association for Information and Image Management
1100 Wayne Avenue, Suite 1100
Silver Spring, Maryland 20910
301/587-8202

1 of 1

Conf-940449--11

PNL-SA-22987

ENVIRONMENT RECONSTRUCTION FOR ROBOT NAVIGATION

S. Bohn
E. Thornton

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.

April 1994

Presented at the
SPIE'S International Symposium
on OE/Aerospace Sensing Conference
April 4-8, 1994
Orlando, Florida

Prepared for
the U.S. Department of Energy
under Contract DE-AC06-76RLO 1830

Pacific Northwest Laboratory
Richland, Washington 99352

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED *JP*

ENVIRONMENT RECONSTRUCTION FOR ROBOT NAVIGATION

Shawn Bohn and Erin Thornton
Applied Physics Center
Pacific Northwest Laboratory,*
Richland, Washington 99352

ABSTRACT

The United State Department of Energy (DOE) is facing a large task in characterizing and remediating waste tanks and their contents. Because of the hazardous materials inside the waste tanks, all of the work must be done remotely. The purpose of this paper is to show how to reconstruct an enclosed environment from various scans of a Laser Range Finder (LRF). The reconstructed environment can then be used by a robot for path planning, and by an operator to monitor the progress of the waste remediation process. Environment reconstruction consists of two tasks: image processing and *laser sculpting*. The image processing task focuses first on reducing the quantity of low-confidence data and on smoothing random fluctuations in the data. Then the processed range data must be converted into an XYZ Cartesian coordinate space, a process for which we examined two methods. The first method is a geometrical transform of the LRF data. The second uses an artificial neural network to transform the data to XYZ coordinates. Once an XYZ data set is computed, *laser sculpting* can be performed. Laser sculpting employs a hierarchical tree structure formally called an octree. The octree structure allows efficient storage of volumetric data and the ability to fuse multiple data sets. Our research has allowed us to examine the difficulties of fusing multiple LRF scans into an octree and to develop algorithms for converting an octree structure into a representation of polygon surfaces.

1. INTRODUCTION

The ability of a robotic manipulator to interact and operate safely within a hazardous environment is essential for performing any prescribed task in that environment. It is the knowledge about this environment that provides the key to solving the navigational issues for the remote manipulator. However, the ability to provide an accurate three-dimensional (3D) representation that is compact and structured for a navigational controller is a nontrivial task.^[1] Remote sensor systems generally collect massive amounts of data points. One major stumbling block is the compression of this volumetric data without loss of accuracy. Without compression the data storage requirements are quite large (i.e., terabytes). We examined this storage issue and issues related to data processing after data collection. Finally, to distinguish our work from others such as Ferrie et al.,^[1] we have decided to focus on modeling the complete environment and not a specific object, which is a key point of this paper.

In reconstructing an environment from multiple sets of sensor data, we address three issues. The first issue is to reduce or smooth noisy and low-confidence data obtained from the sensor. The second is to use the rectified sensor data to generate a 3D computational description of the environment. Finally, the third is to produce a representation of the structured environment for the navigational control system.

The sensor or "camera" used for our experiments was a Perceptron scanning laser ranger finder (LRF).^[8] This LRF has a maximum range of 10 meters and collects images of 256 x 256 data points. For each scene scanned, two files are generated: one contains range values and one contains intensity values. The values of the sampled data have a maximum precision of 12 bits (i.e., 0 - 4095 integer values). For the purpose of our research, we relied on certain manufacturer settings for generating the image data (i.e., square pixels). However, this camera inherently had certain aberrations based on the environment or entities within the environment. Hebert et al.^[6] recognized that this manufacturer's camera had problems with mixed pixels and range/intensity crosstalk. We present the approaches used to combat these problems and to reduce the amount of low-confidence data.

Although not explicitly mentioned by Hebert et al.,^[6] conversion from the sensor coordinate frame to a Cartesian coordinate frame is not purely a trigonometric transformation. This transformation of sensor data is described in section 2.4.1 with the intent of explaining the difficulties of the transformation. Two approaches are discussed: a geometric approach (from the manufacturer) and an artificial neural network. The next process step is the fusion of rectified sensor data to a volumetric representation.

* Pacific Northwest Laboratory is operated for the U.S. Department of Energy by Battelle Memorial Institute under Contract DE-AC06-76RLO 1830.

The problem of 3D reconstruction has been approached by substituting geometric entities that approximate portions of the surface from sensor data.^{[1],[3]} Although this process has the benefit of reducing the data to a mathematical description of the surface, it focuses only on an object. Other authors have taken sensor data and directly fused the data sets into a single object, invariant of translation and rotation.^{[2],[7]} This process has merit for issues of geometric registration and has real potential for volumetric structures. Our research utilizes a volumetric structure called octrees to store and compress data collected from the LRF. Our main impetus for the octree structure was the ability to insert volumetric data from multiple LRF scans. Others^{[5],[10]} have also used octrees for development of 3D models but have not used them in the generation of environment models. We show how we incorporate multiple LRF images into a single octree structure and generate a 3D representation of the environment.

2. IMAGE PROCESSING

2.1 Problem Constraints

Our proposed working environment is a hazardous mix-waste tank. The typical tank has a diameter of 22.86 meters (75 ft) and a depth of 18.29 meters (60 ft). The manipulator requirement is that it must be able to reach into all areas of this 3D space. Each tank may have multiple risers and debris throughout the interior.

To verify our algorithms we collected data from a room whose dimensions were 15 x 30 x 15 ft. Multiple samples were taken throughout the room, and results of scans taken from the four corners are presented in this paper (appendix). All visualization of our work was performed using a Silicon Graphics (SGI) Indigo and Advanced Visual Systems visualization software. Any indication of performance is referenced to this workstation.

2.2 Minimization of low-confidence data

As with any sensor, effects of noise or the general limitations of the sensor cloud the formation of an accurate model. We address some of the problems faced with use of the Perceptron LRF and the algorithms that were used to compensate for noisy data. Hebert et al.^[6] mentioned several problems that arise when using a scanning LRF. The most apparent problems were that of mixed pixels and range/intensity crosstalk. To compensate for some of these problems, the following algorithms, albeit simple, are absolutely necessary for ensuring an accurate and conservative approach for defining the boundary between the environment surfaces and free space. Other researchers^[11] have used other techniques such as maximum entropy methods for reclaiming pixels lost to noise; however, these methods were not used here because of time constraints.

The LRF works by the principle of correlating the intensity of the signal to the time of flight via time phase shift to develop a range value. Based on this concept, a low intensity value provides a low confidence in the resulting range value. Respectively, a high intensity value results in a high confidence in the range value. We constructed a simple clipping algorithm that flags intensity/range pairs as either valid or invalid for integration into the octree. The threshold limit is user-specified based on the type of environment the LRF is scanning. In concert with this clipping algorithm we applied a range clipping algorithm. This is applied directly to the sensor range and performs a front and back clipping based on user-specified data. A common problem with this sensor was the placing of objects either in front of or beyond what is termed the ambiguity interval.^[6]

To address the mixed pixels problem, Hebert et al.^{[6],[8]} proposed applying a median filter throughout the range data. The intent is to smooth out the "spraying" of edge points between the background and the foreground surfaces by selecting the median range value from the data points surrounding the point in question. This produced a smoothing effect on all edges, which creates a fillet or round, depending on the edge type. The mixed pixels were reduced slightly, but the overall effect of softening edges created a dilemma in trying to maintain a conservative description of the environment. To minimize this spraying effect, we used the information of points along the scan in a predictor-corrector approach. The main assumption of the algorithm presumes that a first order lag exists between the foreground and background. Since each image is generated on a scan line by scan line basis, we focus our attention on the pixel ranges on that line. Although this algorithm has some physical basis, Hebert et al. explain that the range values are mostly a function of intensity measurements spanning objects. The following equations depict the algorithm used.

$$P'_{n+1} = P_{n+1} + (P_n - P_{n+1})e^{-\frac{1}{T}} \quad (1)$$

This can then be rearranged as

$$P_{n+1} = \frac{P'_{n+1} - P_n e^{-\frac{1}{T}}}{1 - e^{-\frac{1}{T}}} \quad (2)$$

where $t > 0.0$, $t > 0.0$, P'_{n+1} is the sampled range value, and P_{n+1} is the corrected the range value. By assuming

$$\text{if } |P'_{n+1} - P_n| > Dd, \text{ where } 0 \leq P_n \& P'_{n+1} \leq 4095 \quad (3)$$

where Dd (the threshold value) is specified by the user, the first order time constant equation (2) would be used to predict a new P_{n+1} (predicted range). This algorithm is activated when Dd is exceeded as shown in equation (3). We also improved upon this initial concept by applying the media filter approach along the line to determine the maximum limit for the predicted range value.

The second problem is the range/intensity crosstalk. This effect is caused by the type of laser and detection electronics used. Hebert et al. state that this effect can be reduced by dynamically adjusting the operating range according to the intensity. The camera manufacturer handles this effect by supplying a calibration index for all points in the image, but for only three settings. Our objective at this point was to compensate for the crosstalk without imposing limits on the operational parameters. This was approached by employing an artificial neural network to learn and correct for the crosstalk. This information is presented in the next section.

2.3 Coordinate transformations (rectification)

2.3.1 Geometric approach

A simplified derivation of converting sensor coordinates to a Cartesian coordinate frame is straightforward. The Perceptron system provides the information in a Row, Column, Range format. Figure 1 shows the basic spherical coordinate system. For the purpose of this coordinate system, we chose the y-axis to be the center of the field of view (FOV). The origin is the pivot axis for the nodding mirror and, although the rotating mirror has an offset, we are assuming it is negligible. The coordinates of the point P are given by:

$$x = \rho \cos \theta \sin \phi, y = \rho \sin \theta \sin \phi, z = \rho \cos \phi \quad (4)$$

Figure 1. Spherical Coordinate Frame Setup

The azimuth angle (i.e., the rotating mirror or column value) is proportional to the angle (θ). As per our derivation, the scanner's output does not directly correlate to the angle (ϕ). The elevation angle (i.e., the nodding mirror or row value) is linearly related to (θ). The following relationship is used to compute the corresponding angle within the FOV and then substituted into the equations to convert to Cartesian coordinates. The values from the LRF are denoted as (u, v, r) for row, column, and range, respectively.

$$\alpha = \Delta_\alpha u + \alpha_0, \quad \theta = \Delta_\theta v + \theta_0, \quad \rho = \Delta_\rho r + \rho_0, \quad (5)$$

where

$$\Delta_{\alpha} = \frac{\alpha_{FOV}}{N_{row}-1}, \Delta_{\phi} = \frac{\phi_{FOV}}{N_{column}-1}, \Delta_p = \frac{\text{Ambiguity Interval Distance}}{\text{Max Scanner Precision (Bits)}}, \quad (6)$$

and where the symbols are defined as follows

Δ_{ϕ}	angular increment in $\frac{\text{degrees}}{\text{column}}$ for the azimuth or rotating mirror angle.
Δ_{α}	angular increment in $\frac{\text{degrees}}{\text{row}}$ for the elevation or nodding mirror angle.
Δ_p	proportionality constant for $(\frac{\text{length}}{\text{max bits value}})$
ϕ_0	initial angle for rotating mirror.
α_0	initial angle for nodding mirror.
ρ_0	offset value from camera face.

$$\tan \theta = \frac{z}{\rho \cos \theta}, \quad \tan \alpha = \frac{z}{y},$$

therefore,

$$\tan \theta = \frac{y \tan \alpha}{\rho \cos \theta} \quad \text{since } y = \rho \cos \theta \cos \phi \quad \text{then } \theta = \tan^{-1}(\cos \phi \tan \alpha)$$

Commonly, this transformation from sensor coordinates to Cartesian coordinates is used "as is". Since no part is perfectly manufactured, the variations will cause a deviation between the theoretical and measured values. We found that this simple transformation neglected, for example, the *interval* distance in front of the camera. To solve these problems of crosstalk, manufacturing errors, etc., we examined an artificial neural network for calibration, correlation, and rectification.

2.3.2 Neural rectification approach

As mentioned, there are numerous issues in accurately determining the range value from the row, column, range, and intensity values. Through preliminary research we determined that an artificial neural network (ANN) would provide the capability to learn the correlation of data measured from the physical scene to the sensor data. In addition, we felt that including the intensity information and FOV settings could compensate for crosstalk.

Based on the taxonomy of ANNs, we chose a well defined and understood network referred to as "Backpropagation."^[9] The network we used consists of 10 inputs, 8 hidden layer nodes, and 6 outputs. Figure 2 shows the basic layout of the network, which is fully connected. The input and output parameters are described in Tables 1 and 2, respectively. Each input parameter is normalized with respect to its maximum known limit (e.g., sampled range value / max number of bits).

Figure 2. Neural Network Configuration for Calibration/Rectification

Table 1. Networks Input Parameters

InputNode	Input Definitions
1	Row
2	Column
3	Range
4	Intensity
5	Nodding Mirror Initial Position
6	Nodding Mirror Final Position
7	Step Size
8	Step Delay
9	Divide Ratio
10	Offset

Table 2. Networks Output Parameters

OutputNode	Output Definitions
A	X-Coordinate
B	Y-Coordinate
C	Z-Coordinate
D	X-tolerance error
E	Y-tolerance error
F	Z-tolerance error

Results. We confirmed from our preliminary findings that the neural network was able to learn the correlation from the inputs to outputs to 0.1 percent error. Figure 3 shows the correlation of network inputs to network outputs. The most important aspect of this approach is that only input or output empirical data was considered. All issues with regard to the non-orthogonal axis of scanner parts and crosstalk would be incorporated into the training of the network. We recognize that this was a limited training set, and further studies should include other parameters (e.g., temperature). The next step was to determine if the neural network would be able to reconstruct accurate range data from low intensity data.

Figure 3. Neural Calibration Training Results

3. LASER SCULPTING

3.1 The Laser Sculpting Concept

We approached the environment modeling as a sculptor would, considering a solid cube the size of the environment to be the sculpting block and the LRF to be the sculpting tool. The LRF is deployed into the solid virtual environment and shoots a series of beams within its FOV. Any space between the LRF and each range point (i.e., a point on the surface of the environment) is deemed void space. The process of removing void space from the solid virtual environment is called *laser sculpting*. By repositioning the LRF and taking successive scans, the surface of the environment is determined.

To implement the laser sculpting concept, an octree structure was chosen.^{[10],[14]} An octree is a hierarchical tree structure with cubical nodes. Non-terminal nodes always have eight children, each child being an octant of its subdivided parent cube. Terminal or leaf nodes are flagged as FULL or EMPTY, representing a homogeneous volume.

3.2 Creation of Octrees From LRF Images

Laser sculpting has been defined as the process of removing void space; the functional definition is the process of ray integration into an octree. The generation of the surface is the creation of octants within the octree with the insertion of each ray. Octant volumes are assumed to be FULL until proven EMPTY. The octree grows in definition as each valid Cartesian point is integrated.

An LRF image is integrated by calling the function *oct_insert()* for each valid point, where *p1* is the LRF origin, *p2* is the surface point, and the octant is the octree root. The function *oct_insert()* recursively tests the current octant against the line segment *p1p2*

for intersection. If intersection occurs, octants are subdivided (if necessary) and *oct_insert()* is again called for each child. When octants are subdivided, all eight children are labeled as FULL. If intersection occurs and the current octant is at the resolution limit, the octant is labeled EMPTY. If intersection does not occur in *oct_insert()* the octant returns unaffected. The *oct_insert()* function is listed below:¹

```
oct_insert(p1, p2, octant)
{
    if (octant->flag == EMPTY) return;
    if (oct_intersect(p1, p2, octant) == TRUE) {
        if (size < MIN_RESOLUTION) {
            octant->flag = EMPTY;
            return;
        }
        if (octant->child == NULL)
            octant->child = oct_create();           // mallocs 8 FULL children
        for (i=0; i<8; i++)
            oct_insert(p1, p2, octant->child[i]);
    }
    return;
}
```

The *oct_intersect()* function returns TRUE if the line segment *p1/p2* intersects the octant; otherwise it is FALSE. To implement the *oct_intersect()* function, a modified Cohen-Sutherland line clipping algorithm was employed^[12]. Typically, this algorithm is used to clip lines to a 2D window or 3D viewing volume. A character code is used for each end point of the line segment. The first six bits of the code are used to classify the point as shown below:

```
bit1 = 1 if x < xmin, otherwise 0
bit2 = 1 if x > xmax, otherwise 0
bit3 = 1 if y < ymin, otherwise 0
bit4 = 1 if y > ymax, otherwise 0
bit5 = 1 if z < zmin, otherwise 0
bit6 = 1 if z > zmax, otherwise 0
```

Each min and max value represents a plane orthogonal to one of three principal axes. Once the end points are encoded, the objective is to test for trivial intersection or non-intersection with the octant. If neither condition is found, the line segment is subdivided. The endpoints of a new line segment are encoded and retested for intersection. Ultimately one of the trivial cases will occur. The source code for *oct_intersect()* and *oct_encode()* follows:

```
Boolean oct_intersect(p1, p2, octant)
{
    char          code1, code2;           // 6-bit codes set by oct_encode()
    Point         new_p1=p1;

    code2 = oct_encode(p2, octant->origin, octant->size);
    if (!code2) return TRUE;              // end point inside octant
    for (;;) {                            // forever
        code1 = oct_encode(new_p1, octant->origin, octant->size);
        if (!code1) return TRUE;          // point inside octant (intersection)
        else if (code1 & code2) return FALSE; // trivial reject (no intersection)
        else if (code1 & BIT1) new_p1 = oct_plane(new_p1, p2, X, Xorigin);
        else if (code1 & BIT2) new_p1 = oct_plane(new_p1, p2, X, Xorigin+size);
        else if (code1 & BIT3) new_p1 = oct_plane(new_p1, p2, Y, Yorigin);
        else if (code1 & BIT4) new_p1 = oct_plane(new_p1, p2, Y, Yorigin+size);
        else if (code1 & BIT5) new_p1 = oct_plane(new_p1, p2, Z, Zorigin);
        else if (code1 & BIT6) new_p1 = oct_plane(new_p1, p2, Z, Zorigin+size);
    }
}
```

¹ Some exception handling the variables have been removed for simplicity.

```

char oct_encode(pt, origin, size)
{
    char code = 0;
    if (pt->x < origin->x)          code |= BIT1;
    if (pt->x > origin->x + size)    code |= BIT2;
    if (pt->y < origin->y)          code |= BIT3;
    if (pt->y > origin->y + size)    code |= BIT4;
    if (pt->z < origin->z)          code |= BIT5;
    if (pt->z > origin->z + size)    code |= BIT6;
    return code;
}

```

3.3 Combining Multiple Images

Combining multiple images is made easier with octrees. This is due to our main assumption that LRF positions are defined perfectly in space. Although this assumption is not realistic for practical work, it supplied a basis for our current work, which does not address the registration issues of rotation and translation. However, merging LRF scans is trivial with the octree as long as the data points are accurate and precise. The process begins with one octree maintained as the master octree. When a new scan is taken, its valid data points are integrated into a separate single octree. The master and single octrees can then be merged by simultaneously traversing both trees and performing the equivalent of a Boolean OR on the sub-octants. The master octree is defined as the structure containing the fusion of multiple LRF scans or the environment model.

3.4 Octree Display and Navigation

A visualization of the environment model is necessary for manipulator path planning and for locating new "camera" positions where additional scans could be taken to improve the model. Furthermore, the creation of the environment model (master octree) is an incremental process. Therefore to view the octree from arbitrary positions, we need to move through and around the octree model. The ability to visualize the environment model with such movement requires an interactive display.

To achieve an interactive display, rather than projecting the octree as a quadtree for display purposes (as done traditionally),^[13] a triangle list of the entire octree can be generated from starting node to leaf or minimum-resolution nodes.² If the triangle list is small enough, a graphics workstation can allow an interactive display (for our graphics workstation this number is about 100,000 triangles). Then navigation through the model does not require re-traversal of the octree until the octree itself has changed or some display parameter has changed, such as the resolution of display or the starting octant. The key to interaction is reducing the number of triangles representing the octree. This can be done by displaying sub-trees or partial trees, by lowering the resolution of the octree display, and by using improved octree-to-polygon methods as described below.

3.5 Octree-to-triangle list methods

The objective is to achieve displays of high resolution and low number of triangles, both to improve octree display interaction and to minimize the number of triangles that need to be sent to the navigational controller. Reducing the triangle list by improved octree-to-polygon methods is critical. Presented below are three methods for efficiently generating the triangle list: 1) a simple octree-to-triangle list method, 2) a method that eliminates duplicate and hidden triangles, and 3) a method that also combines coplanar and adjacent triangles. Each method produces the triangle list directly from an octree.

3.5.1 Method 1—simple octree-to-triangle list

To produce a triangle list of an octree is to traverse the tree and—for every leaf node of a specified type—add 12 triangles to the list of triangles (two triangles for each face of the cube). Because the resolution can be varied, there is an issue of what defines the surface. To provide a conservative estimate of the surface (all areas are presumed solid until proven void) at a lower resolution than that of the octree, we must decide whether or not to show partially filled nodes. Figure 4 shows EMPTY left octants and FULL right octants. If the display resolution were the size of the parent and EMPTY octants were being displayed, then the parent

² Throughout the text we will use triangles and polygons interchangeably.

would not be displayed. Display of the parent node in this case would indicate a "safe" area for the manipulator, which would not be true since it is partially full. If, however, FULL octants were being shown, the parent node would be displayed because that would indicate that the area is not "safe" for the manipulator. Hence, we "overdisplay" if drawing FULL octants and "underdisplay" if drawing EMPTY octants.

Figure 3. Partially FULL Octant

3.5.2 Method 2—eliminating duplicate and hidden triangles

The second method of octree-to-triangle list generation eliminates duplicate triangles and hidden triangles from the list. Two octants of the same type would produce two duplicate triangles for their shared surface by Method 1. Edges completely surrounded by other displayed edges should also never be added to the triangle list. To generate the triangle list, we call the *oct_surface()* function (listed below) with our starting octant. The function *oct_surface()* recursively traverses the octree. When leaf nodes are encountered (or the desired resolution is reached), we examine the equal adjacent neighbor for all six sides.^[14] The function *draw_edge()* determines what triangles to add to the list.

```
oct_surface(octant)
{
    if ((octant->child == NULL) || (at_desired_resolution)) { // at leaf or resolution
        if (octant->flag == DISPLAY_FLAG) {
            for (side=0; side<6; side++) {
                neighbor = get_neighbor(octant, side);
                draw_edge(neighbor, octant, side);
            }
        }
    } else {
        for (i=0; i<7; i++) oct_surface(octant->child[i]);
    }
    return;
}
```

To describe the *draw_edge()* function, we refer to Figure 5. The parameter "side" refers to the side of the octant on which the neighbor may be found. Three cases are handled by the function:

- Case 1: If the neighbor is outside of the root octant, the "side" edge of the octant is drawn.
- Case 2: If the neighbor is a leaf, larger than or the same size as the octant, and of the opposite type, then the "side" edge of the octant is drawn. For example, if Octant 20 is the octant and we are considering the left side, then the neighbor is Octant 1 and the left edge of Octant 21 is drawn.
- Case 3: If the neighbor is equal to or smaller than the octant and not a leaf node, then the neighbor must be traversed, visiting only the opposite-side sub-octants. Opposite side edges of leaves of the neighbor resulting from the traversal whose types are opposite that of the octant are drawn. For example, if Octant 1 is the octant and we are considering the right side, then the neighbor is Octant 2. So we start at Octant 2 and visit all sub-octants in the left direction. When the leaves (octants 20, 21, 22, and 23) are encountered, the left edges of those of opposite type (octants 20 and 21) are drawn.

Figure 5. Adjacent Octant Traversal Case

3.5.3 Method 3—combining co-planar and adjacent edges

We have currently designed, but not implemented, an algorithm for combining triangles that are co-planar and adjacent. For example, surface A in Figure 6 would be generated by Method 2 and require 18 triangles. But by combining co-planar and adjacent edges, Surface B would result, requiring only two triangles. Both surfaces would appear identical with gouraud shading.

Figure 6. Difference of Combining Co-Planar and Adjacent Edges

4. CONCLUSION

This paper has discussed the issues in reconstructing an enclosed environment from LRF images. A new filtering technique was demonstrated incorporating a first order lag between foreground and background to handle mixed pixels. Two approaches were shown to rectify sensor data to Cartesian coordinates. Based on the results of these approaches, the Artificial Neural Network technique showed a distinct improvement, but more testing is required. The concept of *laser sculpting* was introduced. The process of integrating rays using a modified Cohen-Sutherland clipping algorithm was discussed. Although this initial work did not address the issues of registration, the structural advantages of fusing multiple octree structures were explained. Three methods were discussed for developing a polygon list to be used by a navigational controller.

5. FUTURE WORK

We anticipate three work tasks that are needed before the environment modeling process could be considered for application to a practical field situation. These are:

- Expand the breadth of the neural rectification technique.
- Incorporate geometric registration procedures for the octree structure.
- Develop procedure for minimizing the number of polygons extracted from the octree structure.

6. ACKNOWLEDGMENTS

I would like to thank Byron Brenden for whose patience and use of the LRF made a portion of this research possible.

7. REFERENCES

- [1] F.P. Ferrie, "Recovery of Volumetric Object Descriptions From Laser Rangefinder Images," *Compute Vision-ECCV 90*, First European Conference on Computer Vision, 23-27 April 1990, pp. 387-396, Springer-Verlag.
- [2] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, "Surface Reconstruction from Unorganized Points," *Computer Graphics*, 26, 2, July 1992.
- [3] S. Sclaroff, Alex Pentland, "Closed-Formed Solutions for Physically-Based Shape Modeling and Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 715-729, July 1991.
- [4] I. Moring, "Acquisition of Three-Dimensional Image Data by a Scanning Laser Range Finder," *Optical Engineering*, Vol.28, No. 8, pp. 897-902, August 1989.
- [5] C. I. Connolly, "Cumulative Generation of Octree Models from Range Data," *Proceedings from International Conference on Robotics*, March 13-15, 1984
- [6] M. Hebert, E. Krotkov, "3D Measurements from Imaging Laser Radars: How Good Are They?", *Image and Vision Computing*, Vol. 10, No. 3, pp 170-178, April 1992.
- [7] G. Blais, M. D. Levine, "Registering Multiview Range Data to Create 3D Computer Objects," Report McGill Centre for Intelligent Machines, September 29, 1993.
- [8] Perceptron Inc., "LASAR Datacamera operation manual," 1993.
- [9] Ben Krose, Patrick van der Smagt, "Introduction to Neural Networks," 1993, fifth edition, internet email: smagt@fwi.uva.nl .
- [10] C.H. Chien, J.K. Aggarwal, "Volume/Surface Octrees for the Representation of Three-Dimensional Objects," *Journal of Computer Vision, Graphics, and Image Processing*, Vol. 36, pp. 110-113, 1986.
- [11] M. Beckerman, F.J. Sweeney, "Restoration and Fusion of Laser Range Camera Images," *Proceedings SPIE Conference on Neural and Stochastic Methods in Image and Signal Processing*, San Diego, CA, July 19-24, 1992.
- [12] James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughs, "Cohen Sutherland Method," *Computer Principles and Practice*, Addison Wesley, Second Edition, 1990.
- [13] Donald Hearn, M. Pauline Baker, " Quadtree display of octrees," *Computer Graphics*, Prentice Hall, 1986.
- [14] Hanan Samet, "Equal Adjacent Neighbor," ACM SIGGRAPH 14th Annual Conference on Computer Graphics and Interactive Techniques, Course 18.

APPENDIX

Six pictures are shown depicting the results of the research. Picture 1.0 is a top view of the four scans from the laboratory. Picture 2.0 is a perspective view of the laboratory with a box in the center of the room. Picture 3.0 depicts void space from the octree of the laboratory. Picture 4.0 is a view of the interior using a front cutting plane showing the box. Picture 5.0 is a depiction of laser sculpting. Picture 6.0 is a view from within the interior using a fisheye lens.

**DATE
FILMED**

7/18/94

END