To the Graduate Council:

I am submitting herewith a master thesis written by Joseph E. Banta entitled "Optimal Range Sensor Positioning During Three-Dimensional Model Reconstruction." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

M. A. Abidi, Major Professor

We have read this dissertation
and recommend its acceptance:

Dan Koch

William R. Hamel

Accepted for the Council:

Associate Vice Chancellor
and Dean of The Graduate School

BEST-NEXT-VIEW:

MINIMIZING THE NUMBER OF RANGE IMAGES

FOR

3-D MODEL RECONSTRUCTION

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Joseph E. Banta

December 1996

ACKNOWLEDGMENTS

I primarily wish to express my appreciation to Dr. Mongi Abidi. As my advisor and the chair of my graduate committee, Dr. Abidi provided a tremendous amount of encouragement and time in my behalf during these two years of graduate school. I would like to express my appreciation to Dr. W. R. Hamel and Dr. D. B. Koch for taking the time to review my compilation of this research. I would also like to thank the United States Department of Energy for the financial support for this research. Dr. Yu Zhien and Xiao Wang should receive credit for their research into the 'best-next-view' topic before my contributions to the project. Additional thanks should go to Todd Smith, Gang Zhang, and Brian Chase for their work pertaining to three-dimensional modeling and visualization. I would also like to thank my wife, Kristye Nicole Spence Banta, for her patience and love.

ABSTRACT

A range image presents direct information of the location of sampled object points in three dimensions. In order for the range information to be useful, it must be registered with an existing model according to the position from which the data was acquired and the orientation of the sensor. Thus, given a finite number of range images in which the majority of an object's surface has been sampled, a prototype three-dimensional model of the object may be reconstructed.

The primary focus of the research described in this thesis has been to design and implement a system capable of automatically reconstructing a prototype three-dimensional model from a *minimum* number of range images of an object. Given an ideal 3-D model, our system iteratively renders range and intensity images of the object from a specified position, assimilates the range information into a prototype model, and determines a sensor position and orientation from which an optimum amount of previously unrecorded information may be acquired. The system tests at each iteration whether the model meets a given threshold of accuracy, at which point reconstruction is terminated. Such a system may find application in the context of robot navigation, manufacturing, or hazardous materials handling.

We have tested the system successfully on several synthetic data streams, and found each set of results to be reasonably consistent with an intuitive human search. The number of views necessary to reconstruct an adequate 3-D prototype will always depend on the complexity of the object or scene and the initial data collected. The models which the system recovers compare well with the ideal models, given a reasonable number of views. Our results illustrate the fact that range information of physical objects can be employed to automatically reconstruct a satisfactory dynamic 3-D model at a minimal computational expense.

TABLE OF CONTENTS

CHAPTER	PAGE
1. INTRODUCTION	**1**
1.1 Problem Definition	1
1.2 Scope of the Current Application	5
1.3 Organization of this Document	7
2. REVIEW OF RELATED WORK	**10**
2.1 Automated Sensor Parameter Planning	10
2.2 Sensor Placement for Robot Navigation	12
2.3 Optical Sensor Placement	13
2.4 Range Sensor Placement	15
2.5 Summary	20
3. RANGE IMAGE ACQUISITION	**22**
3.1 Depth Measurement	22
3.2 Methods of Range Image Acquisiton	23
3.3 Synthetic Range Image Generation	25
4. THE MODELING OF A THREE-DIMENSIONAL OBJECT	**28**
4.1 Representation of Discrete 3-D Data	28
4.2 A C++ Class Library for the Manipulation of Discrete 3-D Data	31

CHAPTER PAGE

4.3 Merging 3-D Data Sets 34
4.4 Visualization of 3-D Models. 37
4.5 Summary. 43

5. DETERMINATION OF THE BEST-NEXT-VIEW. 44
5.1 Method 1: Sensor Positioning According to View History 47
5.2 Method 2: Edge-Based Sensor Placement 49
5.3 Method 3: Targeting the Occluded Voxel Face Centroid 55
5.4 Method 4: Clustering the Occluded Surface Data. 60

6. SYSTEM OPTIMIZATION, INTEGRATION, AND IMPLEMENTATION. 63
6.1 Providing the Sensor with an Unobstructed View of a Feature 63
6.2 Separating Closely-Spaced View Positions 66
6.3 Criteria for Reconstruction Termination. 68
6.4 An Intelligent Collage of Best-Next-View Methods 70
6.5 Software Implementation 72

7. EXPERIMENTAL RESULTS 78
7.1 The Pipe Model 79
7.2 The Mug Model 88
7.3 The Vise Model 93
7.4 The Valve Model 94

CHAPTER PAGE

7.5 The Chair Model 97

7.6 The Hand Model 103

8. SUMMARY AND CONCLUSIONS 109

BIBLIOGRAPHY 114

APPENDIX 121

VITA 124

LIST OF FIGURES

FIGURE **PAGE**

1.1 *The system must engage in the use of whatever sensory equipment is available to populate its knowledge base with any pertinent information* . . 2

1.2 *A digital camera organizes a stream of intensity data into a two-dimensional array of pixels to form an intensity image* 3

1.3 *The reconstruction procedure involves assimilating range data from various positions into a world model* 4

2.1 *Information available from each view position is the visible surface and the void surface*. 19

3.1 *(a) A range image, obtained by anonymous ftp from* peipa.essex.ac.uk, *of a Renault part and (b) the corresponding three-dimensional surface plot* . 23

3.2 *(a) A range image, and (b) an intensity image of a chair model, both rendered using an Inventor model file ported to the SIPP rendering library*. 27

4.1 *A 4×4×4 occupancy grid (3-D bitmap) in which most of the voxels are occupied* 30

4.2 *The range information must be correctly aligned before assimilation into a world model* 33

4.3 *(a) A range image, and (b) a model reconstructed from the surface data contained in and occluded data inferred from the image* 36

4.4 *A 4×4×4 occupancy grid in which the exposed voxel faces are labeled according to their orientation* 38

FIGURE **PAGE**

4.5 *A 128×128×128 occupancy grid rendered at resolutions of (a) 10×10×10, (b) 30×30×30, (c) 50×50×50, (d) 70×70×70, (e) 90×90×90, and (f) 128×128×128* 40

4.6 *The reconstructed 256×256×256 model of a coffee mug, with view positions shown as arrows pointing toward the object center* 41

4.7 *An isosurface rendering of the reconstructed coffee mug model* 42

5.1 *The best-next-view system we will construct may be applied in the contexts of either (a) moving camera, stationary object (MCSO), or (b) stationary camera, moving object (SCMO)* 45

5.2 *(a) Three points on a circle which are equidistant from each other, and (b) the three points on the circle which maximize the sum of the distances from the point to previous points* 48

5.3 *(a) A synthetic range image, and (b) the edge map corresponding to the image* 51

5.4 *The best view position for two surfaces: (a) In order to view two surfaces S_1 and S_2 with surface normals $\mathbf{n}_1$ and $\mathbf{n}_2$, the camera must be in the specified region between the surface planes. (b) The best position to view a surface S is in the surface normal direction $\mathbf{n}$ over the center of the surface. (c) The best position to view two surfaces S_1 and S_2 is thus in the direction that is the combination, $(\mathbf{n}_1+\mathbf{n}_2)$, of the surface normal directions* 52

FIGURE **PAGE**

5.5 *The vector pointing from the center of the image to the brightest pixel average will be the direction we shall use to define the orthogonal surface normal* 53

5.6 *The image up direction* $\boldsymbol{d_{iu}}$ *may be calculated from the view vector* $\boldsymbol{v}$ *and the up direction* $\boldsymbol{up}$ 54

5.7 *(a) A model in which the exposed surface of the occluded model is fairly continuous, and (b) a model in which the exposed occluded surface is composed of several disjoint surface patches* 58

5.8 *Because the surface is evenly distributed about the y axis, the centroid C of the surface is located at y=0.* 60

6.1 *An object which contains a self-occluding concavity* 64

6.2 *A surface normal vector may be rotated about its two angular spherical coordinate axes and consequently ray-traced to determine an unoccluded view direction* 65

6.3 *A situation in which the sensor-placement system has determined a sensor position which is unnecessarily close to a previous sensor position* 66

6.4 *Any two view positions originating from the same point, represented by vectors* $\boldsymbol{V_1}$ *and* $\boldsymbol{V_2}$*, will share a common plane P* 67

6.5 *(a) Because the angle* α *between the previous view position* $\boldsymbol{P_{prev}}$ *and the new view position* $\boldsymbol{p_{new}}$ *is less than the threshold angle* β*, (b) a new value is calculated for* $\boldsymbol{P_{new}}$ *based on a normalized* $\boldsymbol{P_{prev}}$ *and an orthonormal vector* $\boldsymbol{Q_{prev}}$ 68

FIGURE **PAGE**

6.6 *A flowchart of the procedure which the best-next-view system described in this document follows* 73

6.7 *(a) The Control Panel, and (b) the Model Panel of the nextview graphical interface* 76

6.8 *(a) The Image Panel, and (b) the Statistics Panel of the nextview graphical interface.* 77

7.1 *The ideal pipe model* 79

7.2 *The range and intensity images acquired from the first view position for the pipe model* 80

7.3 *A voxel rendering of the recovered pipe model after the assimilation of the first range data set* 80

7.4 *The image resulting from the differentiation of the range image in Fig. 6.2.* 81

7.5 *The range and intensity images acquired from the second view position for the pipe model* 82

7.6 *A voxel rendering of the recovered pipe model after the assimilation of the second range data set* 82

7.7 *The range and intensity images acquired from the third view position for the pipe model* 82

7.8 *A voxel rendering of the recovered pipe model after the assimilation of the third range data set* 83

7.9 *The range and intensity images acquired from the fourth view position for the pipe model* 84

FIGURE **PAGE**

7.10 *A voxel rendering of the recovered pipe model after the assimilation of the fourth range data set, with the largest occluded cluster circled in white* . . 84

7.11 *A voxel rendering of the results of clustering the voxel surface faces in the occluded model of Fig. 6.8* 85

7.12 *The range and intensity images acquired from the fifth view position for the pipe model* 86

7.13 *A voxel rendering of the recovered pipe model after the assimilation of the fifth range data set* 86

7.14 *The ideal pipe model shown with the positions from which the best-next-view system acquired range and intensity images (represented as arrows)* . 87

7.15 *Four graphs illustrating the change in number of surface voxel faces, surface voxels, occluded voxel faces, and occluded voxels for the reconstructed pipe model with the assimilation of each range data set* . . . 88

7.16 *(a) A surface rendering of the reconstructed pipe model, compared with (b) the ideal pipe model* 89

7.17 *The ideal mug model* 90

7.18 *The seven range and intensity images acquired for the mug model* 91

7.19 *The ideal mug model illustrated with the seven view positions* 91

7.20 *(a) A surface rendering of the reconstructed mug model, compared with (b) the ideal mug model* 92

7.21 *The ideal vise model* 93

7.22 *The four range and intensity images acquired for the vise model* 95

7.23 *The ideal vise model illustrated with the four view positions* 95

FIGURE **PAGE**

7.24 *(a) A surface rendering of the reconstructed vise model, compared with (b) the ideal vise model.* 96

7.25 *The ideal valve model.* 98

7.26 *The four range and intensity images acquired for the valve model* 98

7.27 *The ideal valve model illustrated with the four view positions* 98

7.28 *(a) A surface rendering of the reconstructed valve model, compared with (b) the ideal valve model* 99

7.29 *The ideal chair model.* 100

7.30 *The five range and intensity images acquired for the chair model.* 101

7.31 *The ideal chair model illustrated with the five view positions* 101

7.32 *(a) A voxel rendering of the reconstructed chair model, compared with (b) the ideal chair model* 102

7.33 *The ideal hand model.* 103

7.34 *The first four positions from which range data was acquired from the hand model.* 104

7.35 *A voxel rendering of the hand model after four range image assimilations..* 104

7.36 *The eight range and intensity images acquired for the hand model..* 105

7.37 *The ideal hand model illustrated with the eight view positions.* 106

7.38 *Four graphs depicting the size of each model as reconstruction progresses.* 106

7.39 *A voxel rendering of the reconstructed hand model after the assimilation of eight range views.* 107

7.40 *(a) An isosurface rendering of the reconstructed hand model, compared with (b) the ideal hand model.* 108

FIGURE **PAGE**

8.1 *(a) View 1 is acquired from the front of a thin surface, and View 2 is acquired parallel to the thin surface; (b) the image from View 1, and (c) the image from View 2* 110

CHAPTER 1

INTRODUCTION

1.1 Problem Definition

In the event of an intelligent system encountering an unknown object or environment, the system inevitably will be required to reconstruct a prototype or model of the object containing knowledge organized in such a way that the information presented by the model is more suitable for use by the system than information directly extracted from the object. The system must engage in the use of whatever sensory equipment is available to populate its knowledge base with any pertinent information (e.g., color, frequency, location, velocity, etc.) of the object's properties or status (see Fig. 1.1). That information must then be transformed to fit its scope relative to some calibrated frame of reference and fitted for use so that its form allows for more efficient inspection by the system.

For instance, let us briefly consider the case in which an industrial waste disposal team has been assigned to dismantle a facility contaminated with some form of hazardous waste. The team must use all faculties available to acquire information regarding the layout of the facility and the locations of any potential hazards. This information may include any knowledge given *a priori,* such as a schematic of the facility layout or a laborer's first-hand experience there. More likely, the team will be required to insert a system capable of acquiring current information from multi-faceted sensory input, with which the system can construct a working model of the facility allowing tele-operated disposal machinery to be maneuvered within. The model constructed by the system should be accurate and robust in order to prevent injury to equipment or personnel.

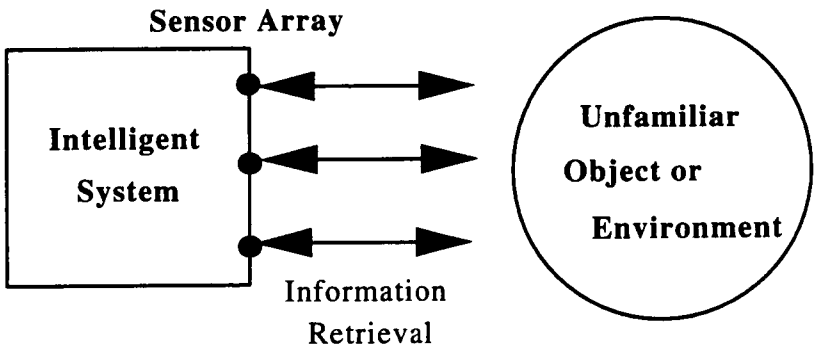


Figure 1.1: *The system must engage in the use of whatever sensory equipment is available to populate its knowledge base with any pertinent information.*

Specifically in this document we shall consider the case in which we would like to reconstruct the *best possible* three-dimensional model of a static object. This will involve acquiring 3-D or range information from a number of locations around the object, consequently transforming it into (aligning it with) the three-dimensional world coordinate system, then updating the model with the new information. The updated model should contain every piece of pertinent information. For this to be possible, we must acquire and manipulate the data as efficiently as the environment will allow. When practical, we will extract multiple sets of usable information from one set of sensory data, or fuse multiple sets of sensory data to obtain an optimally organized set of usable information.

The sensor which we shall primarily use in this research is a *range camera.* A range camera may be seen as being much like an ordinary digital camera, which acquires a stream of *intensity* data and organizes it into a two-dimensional array of pixels to form an *intensity image* (see Fig. 1.2). Rather than intensity data, however, the range camera returns a stream of *range* or *depth* information for each pixel. Thus for each pixel we will have a value (between a pre-specified maximum and minimum) which will represent the distance from the sensor center to the point in space that each pixel represents. This concept will be explained in greater detail in Chapter 3.

Our concern with the range sensor will be to position it such that it will obtain the greatest amount of previously unknown information regarding the object or scene so we will have as complete a 3-D model as possible. This issue will from this point be referred to as the *best-next-view* problem. We define the *best-next-view position* as the next camera pose which will extract the greatest amount of unknown scene information. The ultimate product of a system which solves the best-next-view problem will be a list of a minimum number of sensing operations which will completely reconstruct a model of an object or scene (see Fig. 1.3).

It is possible to implement 3-D reconstruction with edge or silhouette information from two-dimensional intensity data, but depth information from range data will offer much more direct extraction of the type of information we require. In the case of a range sensor, the best-next-view system will acquire 3-D data and assimilate it into a prototype world model, then determine from the current data the sensor location which will potentially extract the greatest amount of new information. The application of a range sensor to the best-next-view problem will offer both an accurate and reasonably cost effective tool for the acquisition of 3-D data.

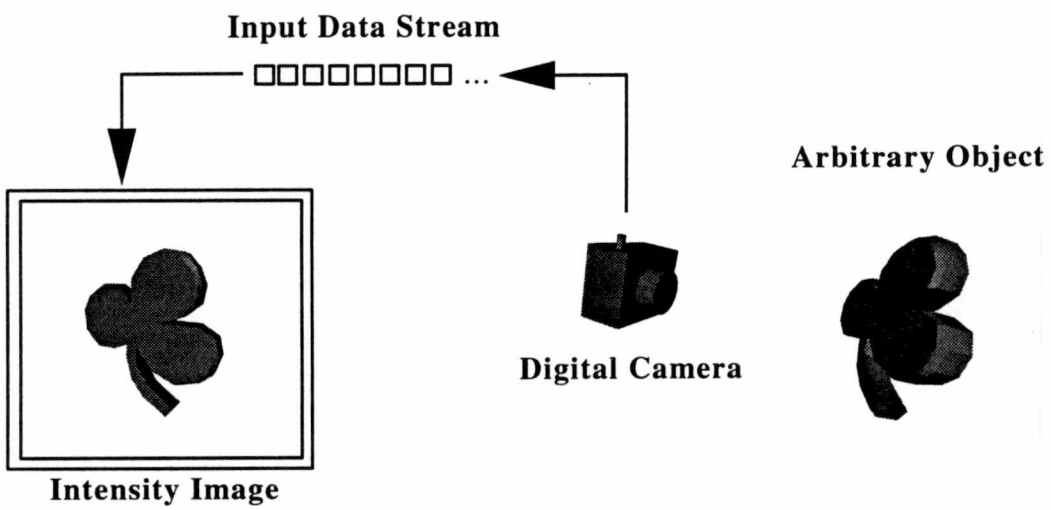


Figure 1.2: *A digital camera organizes a stream of intensity data into a two-dimensional array of pixels to form an intensity image.*

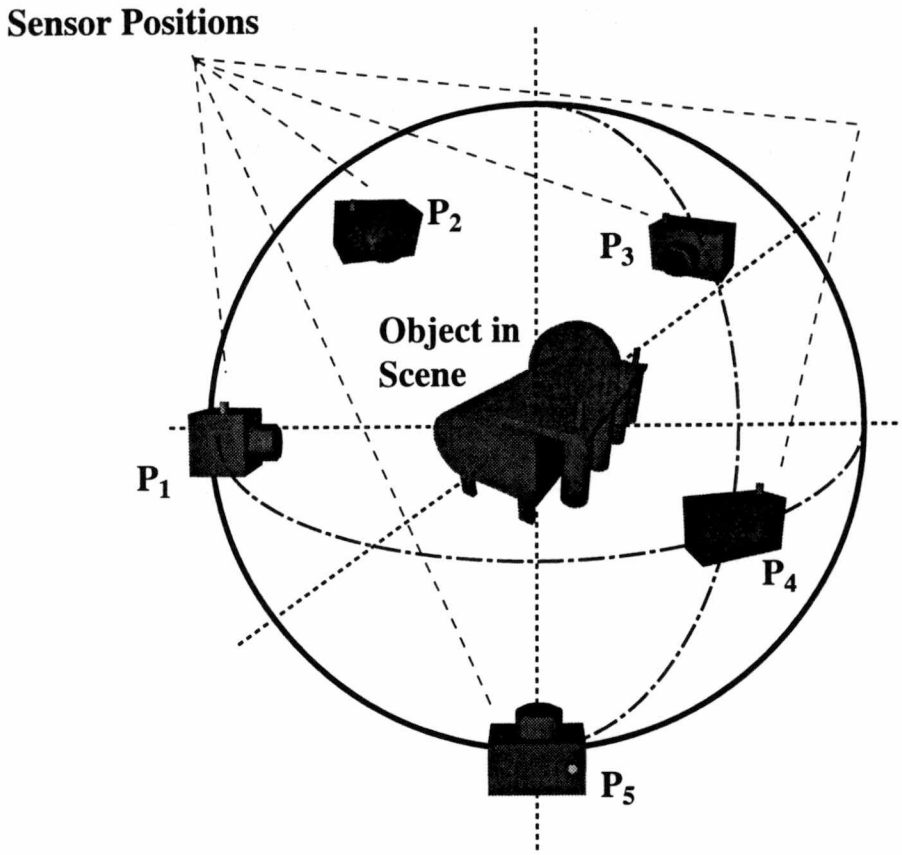


Figure 1.3: *The reconstruction procedure involves assimilating range data from various positions into a world model.*

Determining the location of the best next camera position has always been an important issue in the areas of machine navigation, object recognition, and automatic inspection and manipulation [5]. It is also one of the most difficult issues to resolve in those areas. Although there are many possible sensor positions, the sensor is able to acquire accurate and complete information from only a fraction of them. It is these few locations which we will attempt to find so they may be considered as candidates for the best-next-view position.

1.2 Scope of the Current Application

In this section we will define the restrictions we shall place on any algorithm we develop to solve the best-next-view problem, and make any further definitions which will be necessary in later discussions. Primarily, the system which we wish to develop should be capable of acquiring two-dimensional arrays of range information given a sensor position and orientation. The system should also be capable of transforming the range data into 3-D world coordinates and assimilating the new information into an existing 3-D prototype model. The system must then inspect the updated 3-D model and determine, using a pre-specified set of decision rules or heuristics, the next range sensor position and orientation which will reveal the greatest quantity of unknown 3-D information.

In the process of acquiring range images of actual physical objects, one may expend a large amount of time and effort simply placing the sensor in the specified position. An additional problem that could occur is that once the sensor is positioned, we cannot be absolutely certain of our accuracy without a good amount of additional calibration. Of course, the camera position could be determined after an image has been obtained by finding common features in previous images and matching three or more points in space. This presents somewhat of a dilemma, considering the fact that we must make sure to correctly align multiple coordinate systems *more than once*.

In this research, the system will internally survey an ideal 3-D model of the object and acquire both range and intensity images of it with a minimum of effort. Keeping in mind that the focus of this research is not on range camera calibration, the logical decision would be to keep all range data acquisition internal to the system (at least during the research and development phase of this project), minimizing both computational expense and calibration error. Thus the developed system should take as input a model of a 3-D object and produce a range image (and optionally an intensity image) which may then be registered with a world model.

We should also consider the number of degrees of freedom which will be allowed to the algorithms. In other words, we should determine whether it will be necessary to place certain restrictions on the positions or orientations which will be returned by the algorithms. One restriction which might be considered intuitive when examining a 3-D object is that we cannot place the range sensor *inside* the object. How does one restrict a computer algorithm such that locations inside a model are not considered, when the model has not yet been reconstructed? Also, once the sensor has been positioned, we must be sure that it is oriented such that at least a large portion of the model is visible. A simple solution to both of these problems would be to always place the sensor at a fixed radius from the object center, defined by the user prior to algorithm execution, oriented in the direction of the radius, as in the situation of Fig. 1.3. Of course these restrictions will limit the reconstructive capability of the algorithms somewhat, but they will also greatly simplify the decision which is to be made. Rather than an infinite number of possible points in three dimensions, we now need only consider the positions on a sphere of fixed radius centered around the object. The conclusion is thus that all best-next-view algorithms discussed in this research will return viewpoints located at some pre-determined radius from the object center, and the camera will by default be oriented in the direction of a line radiated from the object center to the returned point.

One issue as yet to be defined in this discussion is exactly what format we shall use for the organization of our 3-D models. This should be considered an issue of great importance in this research, considering the fact that the form of the model will affect the performance of any operations upon it. Possible formats include lists of primitive object types (cubes, cylinders, etc.), 3-D arrays, and octrees. Each format of representation possesses its own advantages and disadvantages which must be considered in determining the best format for our purposes. Because this topic is of such great central importance, it will be discussed in great detail in Chapter 4 of this thesis.

1.3 Organization of this Document

In this section we will briefly discuss the content of the different chapters and sections in this thesis. Chapter 2 will provide a comprehensive review of recent research in areas related to the best-next-view problem. This review will cover research in several sub-groups of sensor planning. These sub-groups include sensor parameter planning, sensor placement for robot navigation, optical sensor placement, and range sensor placement. In this manner, the review should reveal techniques and approaches which might not have been taken into consideration for range sensor placement.

Chapter 3 will focus on the topic of range imaging. In Section 3.1, the concept of depth measurement will be discussed, along with certain other topics pertaining to range imaging. In Section 3.2, we will define a typical method for representing raw range data, and the paradigm which we shall use in this research for range representation. In Section 3.3 we will describe several methods that may be used to obtain a depth map (range image) and cite a number of references where the approaches are discussed in detail. Section 3.4 will describe the application which we have generated to render synthetic range and intensity images from an object model.

In Chapter 4, the focus of interest will be on the modeling of 3-D objects. In Section 4.1, we will describe some of the various formats used for modeling 3-D objects and environments. Each paradigm will be examined for its practicality in such situations as our own, and we will select the one determined to be most suited for our purposes. In Section 4.2 we will present the C++ class library developed specifically for implementation in this context. Our interest is in modeling an object from a series of range images, so this will be the topic of interest in Section 4.3, in which we will describe the algorithm we have implemented for the assimilation of new range information into an existing 3-D prototype. In Section 4.4, we will describe the techniques we have developed for formatting and visualizing the 3-D model in a dynamic rendered context, and in Section 4.5 we will

summarize the complete discussion of Chapter 4.

Chapter 5 will directly address the issue of the best-next-view problem. After introducing the best-next-view problem completely and stating the objective of the system we will develop, we will discuss several approaches we have implemented for determining the best-next-view position. These approaches are completely independent of one another, and take as input a 3-D model of incomplete status and return a position from which a large quantity of new 3-D information may be obtained.

In Chapter 6 we will define the system that is an intelligent combination of the described methods. In Sections 6.1 and 6.2 we will address the optimization of the system. Specifically, in Section 6.1 we will define the method we will use to provide the sensor with an unoccluded view of a feature of interest, and in Section 6.2 we will consider the situation in which the position returned by the system is too close to a previous sensing location. Section 6.3 will define the criteria which will determine when the reconstruction of a 3-D model should be terminated. In Section 6.4, we will use our knowledge of the four approaches described in Chapter 5 to integrate them into an intelligent, autonomous 3-D modeler. Section 6.5 will illustrate the software implementation of this best-next-view system.

Chapter 7 will contain the compiled results from the submission of several 3-D models to the intelligent best-next-view system described in Chapter 6. Each model will be reconstructed at a standard resolution, to be determined in later discussions. The first model will be reconstructed iteratively, with the results of the assimilation of each range data set presented before the next image acquisition. The remainder of the models will be presented in a more concise manner, with only the final reconstructed model visualized for comparison with the ideal model. The fifth model will be reconstructed at somewhat higher resolution, while the sixth model will be reconstructed past the threshold of accuracy set by the intelligent system, to illustrate the diversity and extensibility of the system.

Finally, we will summarize our findings in Chapter 8 and state our conclusions from the system's results. We will make several statements regarding future work, and note potential improvements that could be made to the current system.

CHAPTER 2

REVIEW OF RELATED WORK

In this chapter, we will present an overview of selected documentation in which work pertinent to our own was performed. This overview will be subdivided into several sections, examining certain topics which may be seen as relevant to the best-next-view problem. The first four sections discuss robot navigation, automated sensor parameter planning, model-based sensor placement, optical sensor placement, and range sensor placement, respectively. In the last section we will present a summary of our findings from the researched literature.

2.1 Automated Sensor Parameter Planning

This section will review recent work in the automatic planning of sensor parameters, such as field of view, lighting, shutter speed, etc. Although the general concern of this topic is not quite the same as our own, we will briefly outline some of the documents found that relate most directly to this field.

Hager [30] describes a novel approach to sensor-based decision making based on formulating and solving large systems of parametric constraints. The constraints describe both a model for the sensor and the criteria for optimal sensor planning. The concepts behind this method make it possible to guide the solution refinement process based on task-specific decision criteria. The most obvious advantage of this approach is that it both reduces the amount of work required to reach a decision and increases the likelihood of reaching the correct decision with sparse or incomplete data. Two major weaknesses of the techniques described in this document exist. First, the methods are only suitable for

problems that can be characterized with bounded observation error, known structural models, and relatively few outlier points. Also, it was pointed out that convergence could be quite slow for certain systems of constraints. The system was used to classify certain parameters, including "graspability" (for a robot arm) and shape, of two ball-shaped objects, a compact disc case, and two cylindrical objects.

In the work of Cameron and Durrant-Whyte [16], the authors develop a system that uses the principles of statistical decision theory to determine the optimal sensing locations for performing recognition and localization operations. The system uses a Bayesian approach to incorporate prior object information (including object models or previously acquired sensory data) in selecting sensing locations. The system is optimized such that it will take advantage of whatever type of data is returned by the sensor (and the saliency of the features). The system was tested with a CCD camera mounted on a robot arm. It was noted that further work was required to apply the techniques to a three-dimensional work space, as well as in generating appropriate utility functions for a given task.

Hutchinson and Kak [32] describe an approach to dynamically planning sensing strategies, based on the system's current most reliable information about the environment. The system attempts to minimize the ambiguity which might remain in the updated world model after the sensing operation has been completed. The ambiguity is measured based on the concept of entropy from information theory. Using the pose transformation and object model for each sensing position, the system predicts what features will be observable and finds the sensing operation which minimizes the maximum ambiguity from all the sets of sensing operations. The system was tested on light-stripe range images of a block containing concavities, and returned a set of possible viewing positions which would minimize the remaining ambiguity in the model.

2.2 Sensor Placement for Robot Navigation

An intelligent robot navigating in an unknown environment will manipulate many principles useful to a best-next-view system; both systems must be capable of determining the next position in their environment which will allow further reconstruction of a world model. A robot navigation system, however, will commonly restrict its mapping of the environment to two dimensions (a floor map).

Ali [3] discusses a robotic system capable of navigating a robot along a 2-D floor map and then navigating the arm of the mobile robot in three dimensions. For the 2-D path planning, the robot uses a quadtree generated from a filtered image of the workspace. For 3-D planning, the robot generates an octree from three orthogonal camera positions with respect to a reference point in the workspace. The path is determined from the quadtree or octree using the A^* search algorithm. The system was tested using images of a group of blocks in a static environment, and successfully navigated a path through both 2-D and 3-D environments. The author admits that the constraint on the acquisition of images is unrealistic, but the system was constructed strictly for experimental purposes.

Xie et al. [60] describe an approach to selecting successive viewpoints which allow 3-D models of objects to be constructed by a mobile robot. The 3-D decision is first simplified into a 2-D decision by projecting the partial body models onto the horizontal plane and approximating the projections with line segments. Viewpoints are then chosen to resolve general ambiguities in the scene. The document also describes two related 2-D approaches for planning viewpoints when object models are known *a priori*.

Asada [4] describes a system which builds a map of the environment for a mobile robot from sensory data. A height map obtained from a range image is used to support various tasks such as path planning and landmark recognition. First, the height map is segmented into the regions unexplored, occluded, traversable, and obstacle. The itemized map is then integrated into a global map by matching properties and updating region labels.

Hwang and Ahuja [33] survey previous work on gross-motion planning, including motion planners for point robots, rigid robots, and manipulators in stationary, time-varying, constrained, and movable-object environments. In their paper, several of the general issues of motion planning are explained. In conclusion, recent approaches to the problem of motion planning and possible future research directions are discussed.

2.3 Optical Sensor Placement

In the work of Cowan et al. [21, 22, 23], a system is discussed which determines various optical sensor parameters (camera lens aperture, region in which to place light source, camera position and orientation). The system takes as input models of the vision sensor and its environment and calculates an approximate expected sensor output. The system determines an optimal region for the camera position based on the region of interest on the object model and the initial sensor resolution, field-of-view, and focus requirements. The camera is typically then positioned at the centroid of the region output from the system. In the most recent paper [23], the system was tested on a simple three-dimensional object (a video cassette case). The light source was positioned according to visibility constraints, and the camera position was determined given the target region of interest on the object model.

Kim [35] discusses an approach to model-based object recognition employing multiple views from an optical camera. Emphasis is placed on the determination of camera viewpoints for successive views searching for distinguishing features of objects. The author present the possible relationships between camera and object:

1. Stationary camera, stationary objects (SCSO)
2. Stationary camera, moving objects (SCMO)
3. Moving camera, stationary objects (MCSO)
4. Moving camera, moving objects (MCMO),

and raises two questions regarding the use of multiple images to recognize an object and its

pose:

1. "How can we recognize and determine the orientation of the object with a minimum number of image analyses?"
2. "What should be the next viewpoint of the camera to take the most informative picture?"

In the author's research, the position and orientation of the camera are determined independently; the distance from camera to the object is determined by the size of the object and the feature of interest, while the view direction is determined by the shape of the feature of interest and the presence of occluding objects.

Yi et al. [61] present an optimization approach to sensor and light source positioning, in which geometric measurement and object verification is important. The method defines the optimal light source and sensor positions in such a way that when the sensor and light source are placed, the resulting image produces the minimum variance for the specified measurement. The goal of the vision task is assumed to specified in terms of measurements related to edges. Using a model that takes into account the polarization of light, the contrasts across edges are computed and used to estimate the variance of the required 2-D measurement. The system was tested given a model of an asymmetric block, and the results showed that the formulated problem could be solved by mathematical programming techniques.

Taranabis et al. [55, 56] present the "MVP (machine vision planner) model-based sensor planning system," which synthesizes desirable views of a scene based on geometric models of the environment, optical models of the vision sensors, and models of the task to be achieved. The system attempts to make features of interest visible, inside the field of view, in focus, and magnified as required. Experimental results are presented for the system applied to a robot vision apparatus consisting of a camera mounted on a robot manipulator in a hand-eye configuration. The camera is positioned and the lens is focused according to the results generated by the MVP.

Taranabis et al. [57] also compiled a survey of sensor planning in computer vision. The problem of sensor planning is defined therin as follows:

> "Given information about the environment (e.g., the object under observation, the available sensors) as well as information about the task that the vision system is to accomplish (i.e., detection of object features, object recognition, scene reconstruction, object manipulation) develop strategies to automatically determine sensor parameter values that achieve this task with a certain degree of satisfaction." [57]

Work is reviewed in the field of sensor planning for the task of feature detectability taking the approaches of generate-and-test, synthesis, and expert systems. In summary, the authors describe the work in sensor planning as the initial efforts of researchers to address a problem with many degrees of freedom. It was noted that future work was required for incorporating more realism into the object models, and relaxing some of the constraints made on developed systems.

2.4 Range Sensor Placement

The field of range sensor placement directly addresses the issue of how to determine the next position of the range sensor such that a three-dimensional model updated with the acquired range information will assimilate the greatest amount of previously unknown information. This is exactly the issue which we are attempting to address in this thesis. In this section, we will review work directly relating to this topic.

Some of the first work done in automated range sensor positioning was done by Connolly in 1985 [20]. In this research, Connolly developed two algorithms for determining the best next view using partial octree models. Child nodes of a parent in the octree will belong to one of three types: *empty, occupied,* or *unseen.* The initial view position is completely arbitrary. The first system, the "Planetarium" algorithm, constructs an equally latitudinally and longitudinally sampled sphere around the scene. At each sample point on the sphere, the view of the present model is considered. The point at which the greatest surface area of *unseen* nodes is visible will be selected as the next view

position. The second system developed, the "Normal" algorithm, uses information about the node faces in the octree which are common to both *unseen* and *empty* nodes (the number of ±x, ±y, and ±z faces are counted). A vector is then formed from the larger value for each of the three coordinate axes. The next position of the sensor will lie anywhere on a ray in the direction of this vector emanating from the origin. Testing the two algorithms, the Planetarium algorithm was shown to perform better in the presence of occlusions, but it required on the order of thirty minutes of CPU time on a VAX 11/780 computer, as opposed to the Normal algorithm, which generated a view vector in less than thirty seconds.

Abidi [1] focuses on the use of a vision sensor to perform the volumetric modeling of an unknown object in an entirely autonomous fashion. The use of 2-D vision sensors is examined for the application of closely viewing each local feature of interest, while 3-D sensors are utilized for acquiring global information about the environment. In a proposed automatic modeling system, a 2-D optimization process brings an imaged object into the field of view of the camera, centers the object information on the image plane, and maximizes the image resolution for optimal feature extraction. An entropy-based objective functional is used to model the object or scene at the locations of occluding contours. A local maximum of this function is found to determine the best next view position.

Banta et al. [5] present a range-based algorithm for three-dimensional reconstruction. The proposed system assumes an arbitrary initial sensing position, and uses two-dimensional edge information extracted directly from the most recently acquired range image to determine the best-next-view position. Once each range image has been acquired, the 2½-dimensional range information is transformed directly to the corresponding 3-D coordinates using a transformation matrix calculated from the sensor position and target point. The edge map of a range image should represent discontinuities in a surface, which produce occlusions to the camera's view. The criterion for algorithm termination is set by the percent change in volume of the reconstructed model after the information from each image has been assimilated into it. Results from two synthetic models are presented. The

system performs adequately, but the limits to reconstructing a 3-D model from 2-D information should be recognized.

Maver and Bajcsy [39] describe an automated occlusion-guided view-determining system for scene reconstruction. The system is given *a priori* knowledge of the environment and sensor geometry, and is applied to a light stripe range finder. A characteristic of range images returned from such a sensor is that, of the object surfaces visible to the sensor, only those which are illuminated by the laser (which is generally at some position other than that of the camera) will be ranged; all others will be characterized as occlusions. The system presented returns the next view position from which a complete range image of the surface visible to the camera may be obtained. The system can then compute new scanning planes for further 3-D data acquisition, based on the discontinuities (occlusions) in the most recent range image.

Gerhardt and Hyun [28] pose two questions regarding view planning: (a) how will the object best be viewed; (b) how many views are required to completely verify the object? The authors classify view planning as a subset of sensor planning for systems dealing with vision sensors. In this document, the criteria for view planning is defined by the pixel resolution, surface area, and number of surfaces visible from a viewpoint. The authors propose a system which constructs a geodesic half-sphere around a target object model which contains viewpoints that have been determined to be free from obstructions. A heuristic search algorithm is then applied to find a minimal view sequence. A curved shape and a polyhedral object were propagated through the system to verify it.

Marshall and Roberts [38] address the problems associated with the inspection of geometric features of machined objects using 3-D data acquired by an active stereo vision system. They discuss how three-dimensional data may be processed to produce relevant features, and how it may be inspected to reveal the greatest quantity of unseen data. Having obtained a description of the location and orientation of the observed object from

processed stereo range sensor data, the system determines the most suitable sensor locations given a priori knowledge of an object model (the proposed system is to verify features that *should* be there). To implement this, the system locates each region in space which gives an unobstructed view of features on an object which should be verified. The proposed system is still being constructed, so no results are presented.

In another publication, Maver et al. [40] propose a best-next-view system utilizing the max-min principle as a heuristic. In this document, two systems are proposed. The first system selects from all possible viewing directions the one which gives under the worst-case assumptions for the unknown data the maximal amount of information. The second proposed system computes the viewing directions from which all the data required to accomplish a specified task can be acquired, then selecting from among those directions the one which maximizes the amount of new information. The first strategy can be compared to an algorithm which attempts to obtain at each step the maximal information regardless of the final number of views, while the second strategy takes into account a more global view. Results are presented for a simple geometric environment.

Pito and Bajcsy [44] present a system to automatically acquire a surface model of an arbitrary part and outline the constraints this system places on a solution to the best-next-view problem. This system uses no *a priori* knowledge of the object. Each consecutive range image is required to contain range data of part of the object's surface already scanned, so it may be registered with previously acquired range images. The concept of *positional space* is introduced as a basis for view representation. Two types of information are recovered from range data: the visible surface and the void surface (see Fig. 2.1). The criterion for determining the next view position is set by the void surface area visible from a point. The algorithm is able to determine when all scannable surfaces have been sampled, which allows system self-termination. Illustrated are reconstructed surfaces from a clay bone and a coffee mug.

Sedas-Gersey [47] discusses a system for the automatic positioning of a mobile range sensor. Knowledge of the feature of interest is assumed, including feature position and surface orientation. The process of determining the next sensor pose is divided into two steps. First, a position that satisfies all of the visibility, detectability, accuracy, and density constraints is selected. The second step involves orienting the sensor such the feature is inside the sensor's field of view. The system determines viewpoints based on the contour of the feature surface and the field of view of the sensor. Although a fully developed system is developed, no physical results are presented.

Tarbox and Gottschlich [58] examine the situation in which CAD-based modeling will be used to inspect and verify the expected surface structure of a real object. This involves comparing a model created using intensity or range information of the actual object to a tolerance reference model of the ideal object. Three planning algorithms are presented for finding a set of sensing operations for completely sampling or measuring the exposed surface of an object to be inspected. The first algorithm weights points on the object surface according to "sensing difficulty," and determines viewpoints which can view hard-to-see regions. The second algorithm is similar, but instead of determining the view position from averaged information, view positions are calculated from a lookup table. The third algorithm perturbs a set of initially randomly placed sensing operations in order to find plans that are generally much shorter than could be found by the first two algorithms.

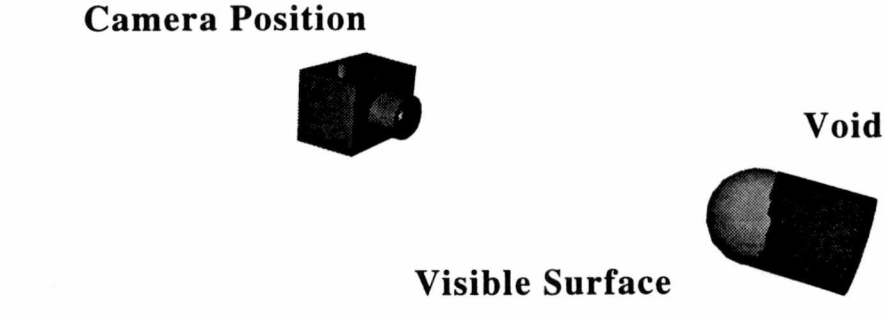


Figure 2.1: *Information available from each view position is the visible surface and the void surface.*

Three sets of results are presented and compared for each algorithm. The first algorithm was found to operate the most quickly, while the second tended to obtain more accurate results, and the third algorithm of course produced the shortest plan.

2.5 Summary

In the previous sections, we have reviewed some of the more significant work related in some way to the best-next-view problem. The research reviewed in automatic sensor parameter planning should provide a variety of heuristics that can be used. Hutchinson and Kak [32] proposed thet idea of minimizing ambiguity in an updated global model in order to proceed with reconstruction. Because the field of robot navigation has the obvious common ground of a sensor array mapping an unknown environment, certain revelations from research into that topic should prove applicable. The approach taken by Asada [4], of segmenting a reconstructed model into the regions unexplored, occluded, traversable, and obstacle, should prove useful for optimal decision making.

Work in optical sensor placement apparently employs many of the same principles as that in range sensor placement. The edge-based sensor placement technique used by Yi [61] to position an optical sensor is similar to the methods used by Banta et al. in [5] and Maver and Bajcsy in [39]. Taranabis et al. [57] formulate the problem of sensor planning with great precision, and this formulation is also applicable to subsets of sensor planning, including range sensor placement. The "Normal" algorithm developed by Connolly in [20] appears to operate on a simple principle, and a variation on this algorithm will be used in Chapter 4. Also, the extraction of two types of information from range data (as performed by Pito and Bajcsy in [44] and shown in Fig. 2.1) will enable optimally efficient use of the range data available.

Many of the principles described in the reviewed documentation, with certain variations and exceptions, will also be adopted in this research. However, there will also be many

original approaches and techniques developed for this thesis. The general expectation is that any method developed will assume a combination of simplicity of implementation and quality of the returned result. Several view-determining techniques will be developed in Chapter 5 according to the standard of this expectation.

CHAPTER 3

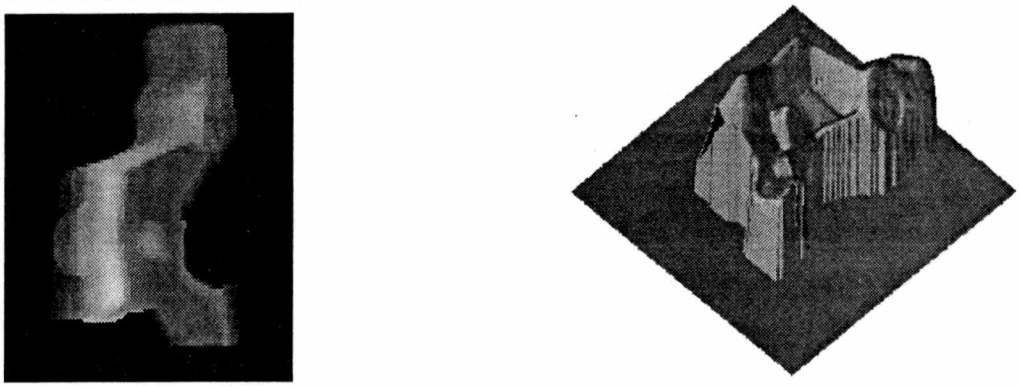
RANGE IMAGE ACQUISITION

3.1 Depth Measurement

A range scanner is a device that senses 3-D positions in a scene and returns an array of distance values. A range image is an $m \times n$ array that contains the radial distance from the sensor center to the point each array element represents (see Fig. 3.1). The range image may be compared with the familiar *intensity image*, which is a two-dimensional description of the object surface as seen when visible light is reflected.

Range is defined as the distance from a point in a scene to a point in a known reference coordinate system. Raw range data, in the form of a range image, only provides us with "2½-dimensional" information—the range image describes only the surfaces ranged from a specific position and orientation as perceived by a particular sensor. In order to extract true three-dimensional scene information, we require knowledge of the camera pose. We define the camera pose as the position and orientation of the camera in world coordinates. Knowing the pose of the camera, we can compute a rigid transformation that will allow us to register newly acquired range data with our world model.

Range data can be represented in Cartesian coordinates, spherical coordinates, or cylindrical coordinates, depending on how the data is acquired and how it is to be used. Range data is normally ordered in a Cartesian Elevation Map (CEM), and is represented in a spherical, sensor–centered coordinate system as (r,θ,ϕ). In this paradigm, r is the distance from the sensor center to the object, θ is the azimuth angle or lateral deflection of the ray, and ϕ is the elevation angle. Both θ and ϕ may vary uniformly between the maximum view angles. Two important parameters of a range scanner are the angular field



(a) (b)

Figure 3.1: *(a) A range image, obtained by anonymous ftp from* peipa.essex.ac.uk, *of a Renault part and (b) the corresponding three-dimensional surface plot.*

of view, which specifies the maximum angular field that the range scanner can cover, and the range field of view, which gives the maximum and minimum depths of field the scanner can detect.

3.2 Methods of Range Image Acquisition

The interest expressed by research and industry in range imaging has increased in recent years, no doubt due in part to the improvement of technology in the areas of data acquisition and parallel data processing. Because the field of range imaging is so broad, we will limit our presentation of related work in that area to several brief citations of work on certain techniques used to acquire range images.

Several ranging techniques employ passive sensors, which project nothing onto the scene in order to determine discrete depth values. Lavest et. al. [36] discuss three-dimensional reconstruction by zooming. From experimentally determined properties of the camera lens, the authors are able to model the optical phenomena which occurs during the zoom lens focal length change and consequently generate a two-dimensional depth map of the sampled scene. Similar to the idea of range from zoom is the concept of depth from focus. In [25], the authors review a suite of methods used in determining the

distances to points in an observed scene by modeling the effect that the camera's focal parameters have on images acquired with a small depth of field. Among other passive ranging techniques is the method of determining depth from stereo images. This is a popular method due to its low cost. In [6], the authors present a method for reconstructing the scene geometry from a pair of stereo images. In [15], the author describes a system which has been implemented which reconstructs a model of its environment from dynamic sets of stereo images. In [48], Sheckar and Chellappa discuss a range finding technique based on a similar principle to the stereo vision concept. The system being researched determines three-dimensional locations of salient points in the environment of a moving camera based on a monocular image sequence obtained by the camera.

Alternatively to passive ranging techniques, we might consider a more accurate active sensing approach. In [14], the authors present a technique for acquiring a range map using color-encoded structured light. In the implementation of this method, a single encoded grid of colored light stripes is used to illuminate the object or scene being ranged. Another instrument that can be used to directly extract 3-D information from the scene is the time-of-flight laser range scanner [17]. Such scanners determine range by measuring the amount of time required to reflect a laser pulse. The phase and frequency of the reflected pulse can be used to construct an intensity image as well as a range image from a single two-dimensional scan. Systems also exist in which pairs of synchronized point scanners optimize triangulation performance. Rioux [45] discusses a system in which two scanners in synchronization allow a linear position sensor to be used for surface topography measurement. The laser range scanner and variations on its concept seem to produce reliable and accurate results. It is the laser range scanner which we will attempt to emulate with our scanning of synthetic models.

More information on range imaging can be obtained by consulting a survey of range finding techniques [34]. Jarvis [34] outlines the methods of contrived lighting range finding, relative range from occlusion cues, depth from texture gradient, range from focus,

range from stereo disparity, range from camera motion, Moire fringe range contours, simple triangulation range finding, and time-of-flight range finding.

3.3 Synthetic Range Image Generation

As stated in the introduction, our goal will be to acquire synthetic range images of an object model and assimilate them into a prototype model. Any of the techniques described in the previous section can be used to acquire a range image. Because we are synthesizing a range scanner, we will be able to acquire near-ideal range images. By emulating the time-of-flight laser range scanner, we can completely measure the depths visible from a view position. For our research, we will assume the range sensor to be at a fixed radius from the object center. This will alleviate the need for a scaling component to the transformation matrix. We will note that this restriction is not *absolutely necessary*. It would be possible to implement a synthetic imager capable of zooming in or out on a particular region of interest. For the purposes of our research, however, we can greatly simplify the process of range image acquisition and the assimilation of the range information into three dimensions if we consider only those points at a fixed radius from the object.

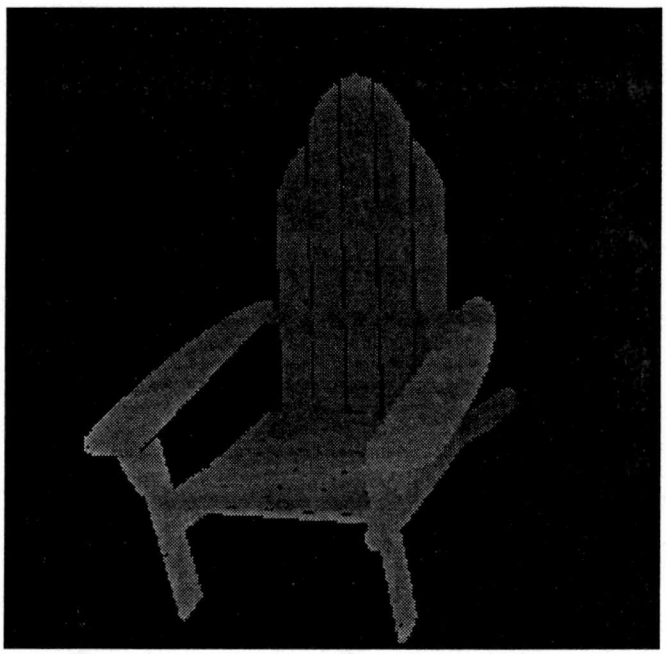
In order to input an object model into a range synthesizer, we will require an exterior format for a three-dimensional model file. A practical and well-known 3-D file format is the SGI Inventor file format [41]. A good deal of documentation is freely available regarding the Inventor format. It was first released in July 1992 by Silicon Graphics, and was specifically designed for the exchange of 3-D modeling information between software applications. It has been used for CAD, chemistry, financial data, scientific visualization, art history, earth sciences, presentation, architecture, animation, and other applications. Created as part of the IRIS Inventor 3-D Toolkit, the Inventor file format is used by a large assortment of 3-D applications and is the basis for the Virtual Reality Markup Language, an

annex to the HyperText Markup Language. The Inventor format essentially creates an object-oriented database, where objects can include groups, files, transformations, labels, materials, drawing styles, lights, at least 18 different geometry types, and so on.

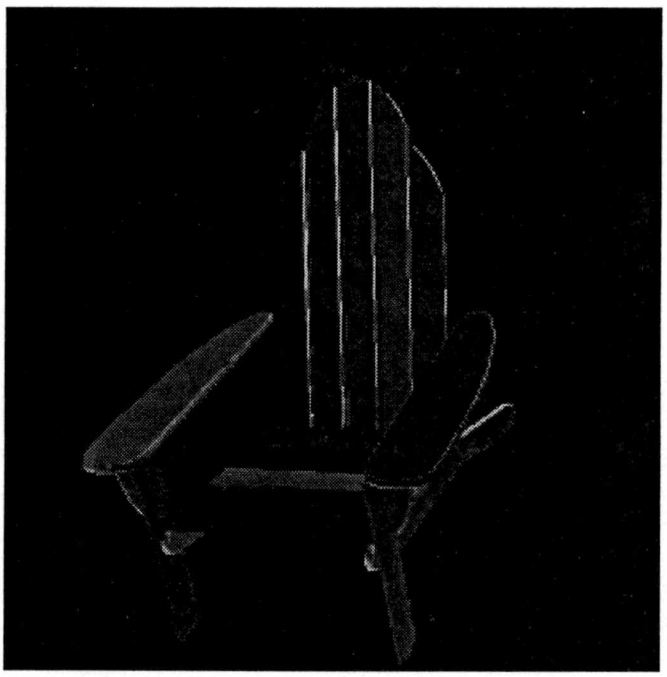
We have developed a software application which will take as a command line argument the name of an inventor file which contains a 3-D model of an object. The application reads the input model file and renders still range and intensity images of the object either from a default view position or a position specified on the command line. Rather than writing the application using the IRIS Open Inventor Toolkit, we chose to use a rendering package: the SIPP (SImple Polygon Processor) rendering library freely available from the Free Software Foundation in Cambridge, Massachusetts [62].

Porting a 3-D model from Inventor into SIPP is actually a fairly simple process. In the Inventor file format, three-dimensional surfaces are represented as a list of connected polygon vertices. In order to render a surface using the SIPP library, one passes the library a similar list of polygons. The only difference in the two surface formats is the ordering of the vertex list. Colored polygons are supported by both packages, allowing color to be incorporated into any surface rendered. Thus, we were able to generate a rendering application for both range and intensity images.

An illustration of the rendering capability of the application is shown in Fig. 3.2. In Fig. 3.2(a) we see a range image of a model of a chair, and in (b) we see the corresponding intensity image. The Inventor model which defined the input to the rendering application was composed of a number of surfaces which could each be broken down into a list of triangular polygons. In this research, we will routinely employ a range image output from this application as input into another application, a function which will translate the 2½-dimensional range data into the 3-D world coordinate system. From the updated model, we will go on to determine the best-next-view position using one of a number of algorithms.



(a)



(b)

Figure 3.2: *(a) A range image, and (b) an intensity image of a chair model, both rendered using an Inventor model file ported to the SIPP rendering library.*

CHAPTER 4

THE MODELING OF A THREE-DIMENSIONAL OBJECT

The modeling of a three-dimensional object or environment may be accomplished using data from one of many available types of sensors. In our research we will be concerned with modeling objects from range data, as discussed in Chapter 3. In this chapter, we will review several of the issues regarding the modeling of a 3-D object. In Section 4.1, the topic of interest will be the format of representation for the 3-D models in this research. We will review several possible formats, and determine the one most suitable for our purposes. Section 4.2 will discuss the C++ class library we have compiled which implements that format. In Section 4.3, we will describe the algorithm we have developed for merging multiple range data sets of an object acquired from different perspectives. The methods we have implemented for visualizing the models will be discussed in Section 4.4, and in Section 4.5 we will summarize our discussion of 3-D modeling.

4.1 Representation of Discrete 3-D Data

Representation is obviously one of the fundamental building blocks of a system working with three-dimensional models. The most practical paradigm of representation for our purposes will be one which provides a terse, compact illustration of the object model. The improvement of hardware facilities for parallel computation will encourage the development of algorithms for computing properties of solids from these representations. Also, with the declining cost of computer memory, explicit and intuitive representation of objects may become increasingly more common.

If we first consider the model format organized by a list of primitives, we may note that if we were to reconstruct such a model, we would be required to recognize the primitive objects within each range data stream. For instance, we might examine the most recently acquired image for evidence of the presence of a cone. In order to find such evidence, we would need to filter the image in some way so that the parameters or elements of the cone revealed themselves (such a process is reviewed in [24]). Without any further investigation, we can see that this line of action would lead us into more calculation than necessary.

If we next consider reconstructing the model as an octree, we will recall that this process is discussed in several instances in the reviewed literature [11, 19, 20, 37]. Therefore several methods have already been defined for reconstructing an octree from range data. Another issue to be considered, however, is whether or not the octree format of representation is indeed practical for use in a real-time application. Our reason for reconstructing a model, after all, is so that we can determine from it the next location from which it would best be viewed. Thus we shall be continually testing the states of nodes (voxels) in the octree so that we can make the decision. In order to access the state of a node in the octree, each parent node in the node's hierarchy must be traversed until a terminal level is reached. In iterative calculation, this could retard the decision process a great deal.

The most intuitive paradigm of representation for a three-dimensional object is the occupancy grid. The occupancy grid is simply a three-dimensional binary array in which each element represents a specific volume in the discrete tessellation. The grid may be viewed as a simple extension of the concept of a two-dimensional bitmap. Just as each pixel (i,j) in a 2-D bitmap can take on one of two binary values, each voxel (i,j,k) in the 3-D bitmap (occupancy grid) can take on a value of 0 (unoccupied) or 1 (occupied) (see Fig. 4.1).

An occupied voxel in the occupancy grid represents a cubic volume within the 3-D environment which is intersected by an arbitrary solid. Complementing the occupied state is the unoccupied voxel state, which obviously describes a cubic volume within the 3-D environment which is not intersected by any object. The resolution of the occupancy grid will be determined at the time of reconstruction; an occupancy grid can be used to construct an object model up to any specified resolution.

Unlike the list of primitives, the occupancy grid cannot represent a continuous surface in the 3-D environment. The simplicity of the binary format restricts the representation of the model to a jagged, uneven illustration of the object(s). Obviously, then, the higher the resolution of the grid, the more realistic the presentation of the continuous object. With higher resolution, however, comes a higher memory requirement and more time required for processing. If we consider a 500×500×500 occupancy grid, the number of voxels in the grid will be 500^3, or 125 million voxels! Thus practicality restricts realism, and we will generally limit the size of any occupancy grids in this research to size 256×256×256 or less (almost seventeen million voxels).

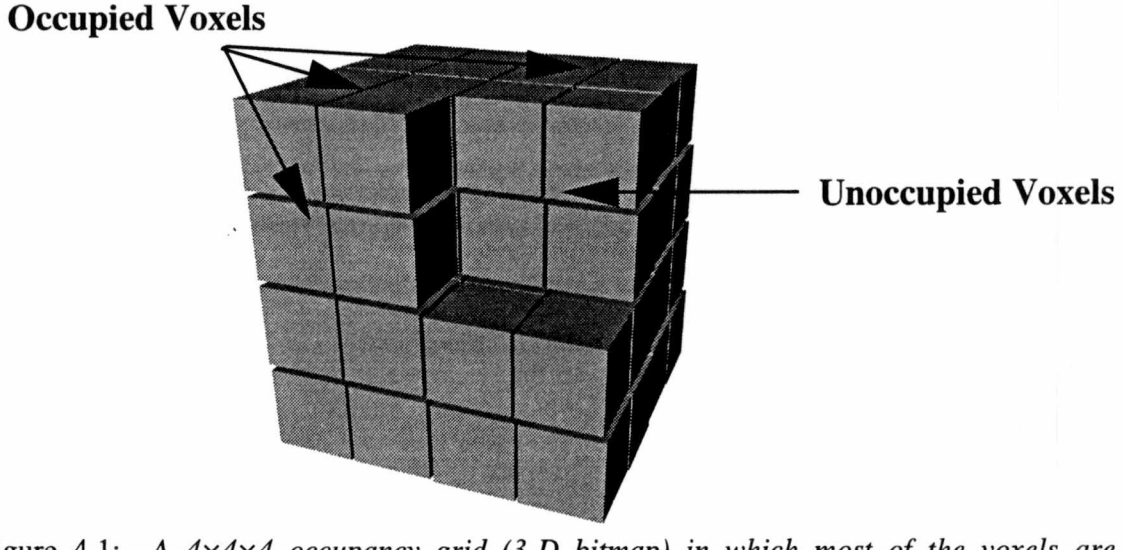


Figure 4.1: *A 4×4×4 occupancy grid (3-D bitmap) in which most of the voxels are occupied.*

The issue of actually recognizing the occupancy state of a specified voxel from the information contained in a range image has been confronted in many instances in literature [7, 26, 31, 46]. The matter of reconstruction is then made simple by both the simple nature of the occupancy grid paradigm and the existence of methods to build such grids from range data. Accessing an element of the occupancy grid is also a simple matter. Because the grid is ordered in the manner that we specify, we need only to search for a certain offset in the voxel array to find the occupancy state of a voxel. Also, an occupancy grid is not limited to resolutions of size $2^n \times 2^n \times 2^n$. Not only do the three dimensions not have to be uniform powers of two; they also do not need to be identical. For example, it would be possible to have an occupancy grid of size $12 \times 145 \times 1$. Thus it seems that the occupancy grid is the most efficient paradigm of representation for our purposes.

4.2 A C++ Class Library for the Manipulation of Discrete 3-D Data

Because so much of this research relies on the paradigm of representation of our three-dimensional models, it would be to our advantage to have a standard 3-D model data type which we could manipulate and operate upon. Such a data type would be of great utility if it could be passed between a reconstruction function and the view-determining functions. We would require the capabilities of voxel access from specified elements, value-setting at specified indices, logical operations, and overall data efficiency.

With these issues in mind, we have created a C++ class library for the manipulation of occupancy grids and octrees. The class declarations for the two classes, including all members and member functions, are listed in the Appendix. The class created for occupancy grids we shall name Volume Grid ("VolGrid"), and the class created for octrees is simply called "Octree."

Class VolGrid is the more complete of the two classes, because it will probably be called on more often. The private members include the x, y, and z dimensions of the grid

and the binary array that contains the grid voxel information. Each element of the array contains one byte (eight bits) of data, and thus each element contains the states of eight voxels. The array is ordered such that the first element contains the states of the first eight voxels in the first x row at the first y and z. Once the row is complete, y is incremented. Once an x-y plane has been traversed, z is incremented. The class is capable of containing a 3-D occupancy grid of any resolution M×N×P.

Class VolGrid contains member functions for returning any of the dimensions, a voxel state at any specified point, or a VolGrid containing a sub-grid of the original grid. A grid of the same size as the original or smaller can be inserted at a specified point into the grid. The class will read or write a file (with an appropriate header). Member functions exist which will occupy or empty individual voxels, a voxel range, or the entire grid. Available logical operations between VolGrids include a logical not, an equality test, an inequality test, and a voxel-wise logical and, or, and exclusive or. A VolGrid may be set equal to another VolGrid or an Octree.

Class Octree is a fairly simple class. Its private members are the power of two that determines the size of the octree and the binary array that contains the octree itself. The binary array is ordered such that the first two bytes contain the states of the largest eight nodes. The nodes can each take on a value of 0 (parent), 1 (unseen/unknown), 2 (unoccupied/empty), or 3 (occupied). If any of the nodes are parent nodes, then the next two bytes available in the array describe the states of the parent's sub-nodes. The tree traversal proceeds until every node in the octree is homogeneous.

Member functions for class Octree include reading and writing files, element access, and dimensionality access. An Octree may be set equal either to another Octree or a VolGrid. The decompression of a 128×128×128 Octree into a VolGrid on a Silicon Graphics Indigo II 150 MHz R4400 work station requires approximately 8.4 seconds; for a 256×256×256 Octree, this time increases to approximately 23.9 seconds (depending on th

complexity of the object). Because there are essentially three possible states for each node (one unoccupied and two occupied states), an Octree can contain two non-intersecting VolGrids. Therefore we have a function which splits an Octree into two VolGrids and a function which combines two VolGrids into one Octree.

The two classes should serve to complement one another. Because the format for the occupancy grid is simple, it will be used the majority of the time for general-purpose applications, such as when we would like to manipulate any part of the volume model. However, the octree paradigm also possess its advantages; because an octree is a volume-wise associative model, certain types of information may be drawn from the node density in the model.

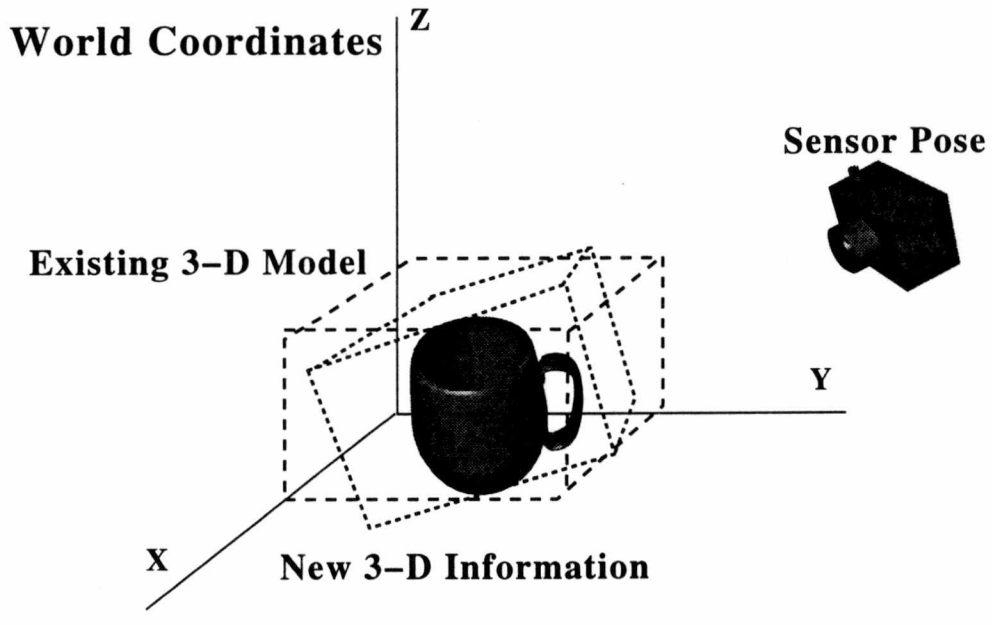


Figure 4.2: *The range information must be correctly aligned before assimilation into a world model.*

4.3 Merging 3-D Data Sets

As previously stated, a new stream of range data may be used to update a three-dimensional model of the environment only if the sensor pose from which the range data was acquired is precisely known (see Fig. 4.2). Although we will assume in our research that the sensor pose is known a priori, it may be calculated by matching object characteristics in the most recent range image with partial characteristics from previous images. The sensor pose can be separated into two components: the sensor's position and the senor's orientation. Thus a transformation which aligns the range data with the world model will have two components, specifically a translation and a rotation. Also, if the range images were acquired at a different resolution or from different distances, the range data being assimilated must be scaled such that each three-dimensional element (voxel) in the updated model will have the same weight and be of the same size.

Thus we must apply a transformation that maps a point (ρ,θ,ϕ) to its corresponding Cartesian coordinates (x,y,z), where x and y correspond to the position in the image plane and z represents the magnitude of the normal from the plane to the point. To reiterate, the transformation from the Cartesian range image to three dimensions is made up of a translation matrix, a rotation matrix, and possibly a scaling matrix. The values which will ultimately comprise the transformation matrix will depend upon the current camera pose relative to the world coordinate system.

Also, in order to reconstruct a 3-D volume of the scene, we need not only to map the range data to three dimensions, but also to identify those points which are occluded by the transformed surface points. To identify these hidden points, we can extrapolate along the rays connecting the range sensor and the detected surface points.

Our objective will be to transform the range data from "2½ dimensions" to three dimensions—determining the actual position in space of each point represented in the range image by calculating the transformation from one coordinate space to another. We can

easily shift a greyscale range image to three dimensions by separating the grey levels into binary bit planes and occupying the voxel corresponding to each pixel and appropriate bit plane. Any remaining transformation will be dependent on the sensor pose.

Because the range images will all be of the same resolution and acquired at the same radius from the object center, neither the translation nor the scale factor component of the transformation will be necessary. Bittar et al. [11] stated that the rigid body transformation between two positions may be formulated by a translation vector $\mathbf{t} = [t_x, t_y, t_z]^t$ and a 3×3 rotation matrix **R**. This is the formulation which we shall adopt. The rotation matrix **R** will have three independent parameters—the Euler angles—θ, ϕ, and ψ, and will be of the form:

$$\mathbf{R} = \begin{bmatrix} \cos\psi\cos\theta & -\sin\psi\cos\theta & \sin\theta \\ \cos\psi\sin\theta\sin\phi + \sin\psi\cos\phi & -\sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & -\cos\theta\sin\phi \\ -\cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi & \sin\psi\sin\theta\cos\phi + \cos\psi\sin\phi & \cos\theta\cos\phi \end{bmatrix}. \quad \textbf{(4.1)}$$

and the rigid transformation **T** may be expressed as:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}. \quad \textbf{(4.2)}$$

t in our case equals $[0, 0, 0]^t$. The angles θ, ϕ, and ψ are simply the angles of rotation about the x, y, and z axes, respectively.

Once we have calculated the transformation matrix, we may proceed with the registration of our new range data. We will begin with a fully occupied *world occupancy grid* and a fully empty *surface occupancy grid.* The *surface occupancy grid* will be comprised of the surface information obtained directly from the range images. We will create the *world occupancy grid* from the combined surface information obtained from the range data and the hidden information which lies occluded from the camera's view behind the ranged surfaces. This concept is similar to that of the visible surface and void surface, developed in [44] and illustrated in Fig. 2.1 (the two techniques were developed independently). An illustration of our approach is seen in Fig. 4.3.

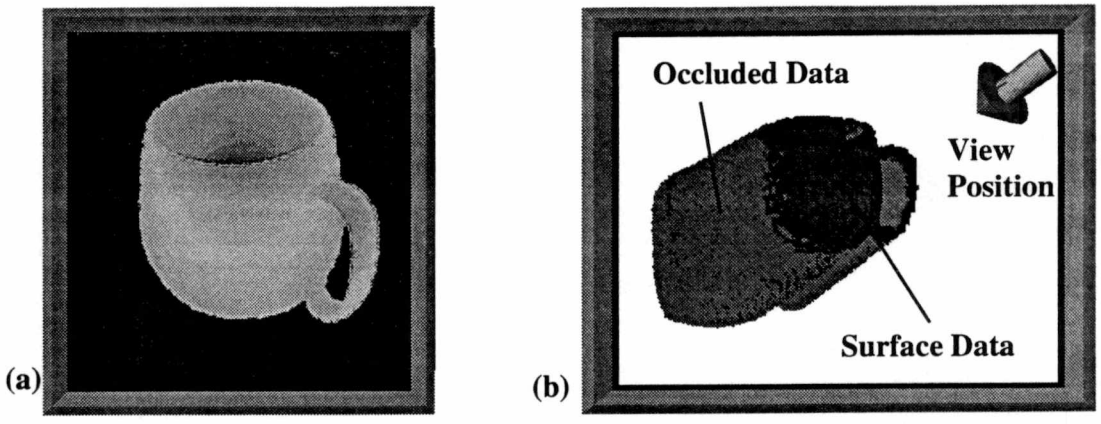


Figure 4.3: *(a) A range image, and (b) a model reconstructed from the surface data contained in and occluded data inferred from the image.*

Our approach will be to iterate through every x, y, and z index in the grid and apply the transformation at each point. From the x, y, and z indices of a point, we will compute the projection of that point onto the image plane. We will then apply one of three rules:

1. If the pixel in the range image corresponding to the current 3-D indices is a point which did not return a range value, that voxel in the world occupancy grid will be empty.

2. If the voxel is in front of the image plane (the projection index is greater than the grey level of the corresponding pixel in the range image), that voxel in the world occupancy grid will be empty.

3. If the projection index is equal to the grey level of the corresponding pixel in the range image, that voxel in both the world and surface occupancy grids will be occupied.

Once we have iterated through every index in these new occupancy grids, we will then assimilate the new information into the existing 3-D database using voxel-wise logical operations. We would like for only the voxels which are occupied in every new grid to be occupied in the updated prototype model. This is accomplished using the logical AND operation. Between two elements α and β, the AND operation is represented as $\alpha \cdot \beta$. Thus, if our existing 3-D model is called *world*, and the new model from range data is called *tworld*, we can update the world model with the operation:

$$world = world \cdot tworld. \tag{4.3}$$

We will treat the surface information somewhat differently. We would like for the voxels which are occupied in any surface grid to be assimilated as part of the prototype surface model. This is accomplished using a voxel-wise OR operation. Between two elements α and β, the OR operation is represented as $\alpha+\beta$. Because range information returned from the edge of an occlusion will often be highy discontinuous, a secondary issue which we must address is assuring that all the surface information which we assimilate into a surface prototype will be valid. This problem can be solved by checking to see if the new surface information is contained in the updated world model. In other words, we will perform a logical AND operation on the surface prototype OR'ed with the new surface information and the updated world model. Therefore, if the existing 3-D surface model is called *surface* and the new model from range data is called *tsurface*, and we have already assimilated the new world information into the world model *world*, we can update the surface prototype with the operation:

$$surface = (surface + tsurface) \cdot world. \tag{4.4}$$

With the updated models we will be able to make intelligent decisions regarding issues such as model accuracy or future range sensor placement. By "subtracting" the surface model from the world model, or emptying out all voxels in the world model which are common to the surface model, we can obtain a model of the volume occluded from the collective camera views.

4.4 Visualization of 3-D Models

In this section we will discuss the techniques we shall use for visualizing the 3-D models. For our purposes, we will consider each voxel in the 3-D bitmap to represent equivalent cubic volumes. In our best-next-view research, we will primarily base our sensor placement decisions on the positions and orientations of the exposed voxel faces. We will define an exposed voxel face as the exposed side of the cube corresponding to the voxel. In

Fig. 4.4, we illustrate a 4×4×4 occupancy grid in which the exposed voxel faces are labeled according to their orientation (possible orientations are +x, −x, +y, −y, +z, or −z). Primarily the +x faces are visible from the example vantage point. Because this is the interpretation of the model on which the best-next-view system will base its decisions, we should be able to visualize the model in this way. More specifically, we should be able to render the exposed voxel faces in the model in order to visualize it.

In current literature, we have found many algorithms for visualizing 3-D data. A large portion of them represented the 3-D model as a surface segmented into optimized triangles, such as the Delaunay triangulation algorithm [9]. One possible discrepancy between this method and ours is that our models are *volume* models. Although there are algorithms which exist which will take as input a 3-D occupancy grid and output a representation of the exterior surface, this method forces interpolation into the representation. Thus a slight imprecision will be introduced so that we can make the object surface more attractive to visualize. Taking into consideration the facts that the original voxel face representation is both a more accurate interpretation of the object model and it is closer to the fundamental format upon which we will be making our decisions, we consider the voxel face

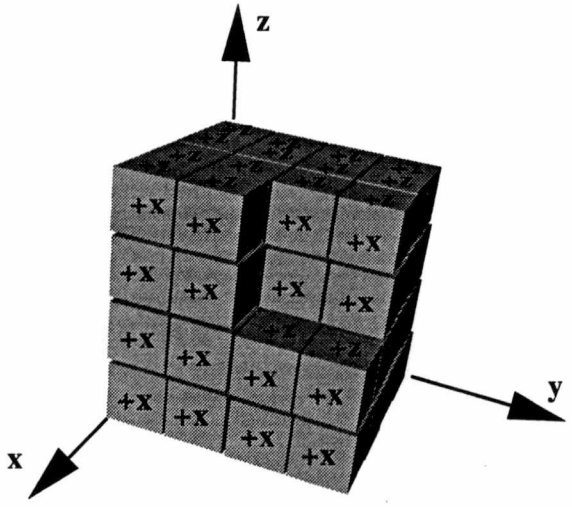


Figure 4.4: *A 4×4×4 occupancy grid in which the exposed voxel faces are labeled according to their orientation.*

visualization to be the most appropriate for our purposes. However, in order to present a visually attractive rendering of a 3-D model, we should also be capable of visualizing an isosurface rendering of the model. This will both broaden the capability of the system and diversify the process of visualization.

All 3-D data visualization in our research has been done on the Silicon Graphics platform. Earlier in this chapter we reviewed the IRIS Inventor file format, which allows the 3-D model to consist of various object types. One of those types is the FaceSet, a surface representation in which the vertices of each polygon and the number of vertices in each polygon are specified. This format of representation will prove convenient for our purposes. We will iterate through the occupancy grid to find occupied voxels which have adjacent voxels which are not occupied. We have written a C++ application which will take as a command line argument the name of a file which contains an occupancy grid and write to standard output a file of IRIS Inventor format describing the surface of the occupancy grid. One may specify on the command line the color of the surface and the maximum resolution of the output grid. Also optional are the three coordinate axes and a uniform wire frame sphere surrounding the grid, to give the user an idea of the scale and orientation of the grid, respectively.

The input file may contain a single occupancy grid, or it may be a two-model file, as described in the previous section. For each occupancy grid, the application iterates through the full x, y, and z dimensions of the grid and tests the state of each occupied voxel's neighbors. If the voxel's ±x, ±y, or ±z neighbor is unoccupied, or if the voxel is on the verge of the grid, then the corresponding voxel face is added to the list of faces to be drawn. When we are rendering two occupancy grids, we will simply check to see if the corresponding neighbor in either occupancy grid is occupied. To minimize the number of polygons we need to render, we will group adjacent faces into one polygon; e.g., if two adjacent voxels have exposed +x faces, then we will simply render one polygon that is twice as long. In Fig. 4.5 we have organized four still snapshots of one of our occupancy

grids piped into an Inventor Examiner Viewer. The four examples are all the same model rendered at a different resolution. The original model was a 128×128×128 grid, which is the resolution rendered in Fig. 4.5(d). The remaining grids serve to illustrate the effects of lesser resolution on the appearance of the visualized model.

We will note that the model file will contain the positions from which each view was obtained. We may also incorporate these positions into our visualization. A simple and intuitive representation of a view position would be as an arrow pointing toward the object center. In Fig. 4.6, we see a voxel rendering of a 256×256×256 reconstructed model of a coffee mug, surrounded by a sphere of the radius from which each range/intensity view was acquired. On the view sphere are seven arrows pointing toward the object center representing the seven positions from which the views were acquired. Also contained in

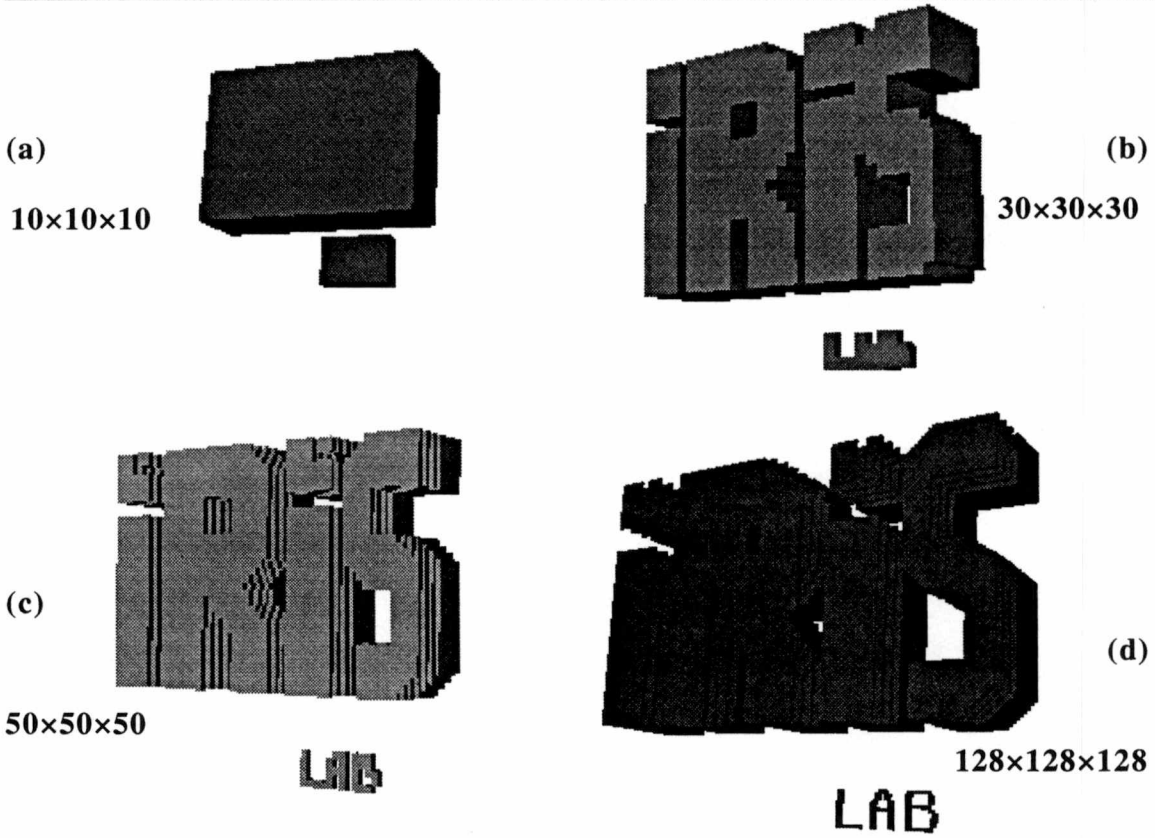


Figure 4.5: *A 128×128×128 occupancy grid rendered at resolutions of (a) 10×10×10, (b) 30×30×30, (c) 50×50×50, and (d) 128×128×128.*

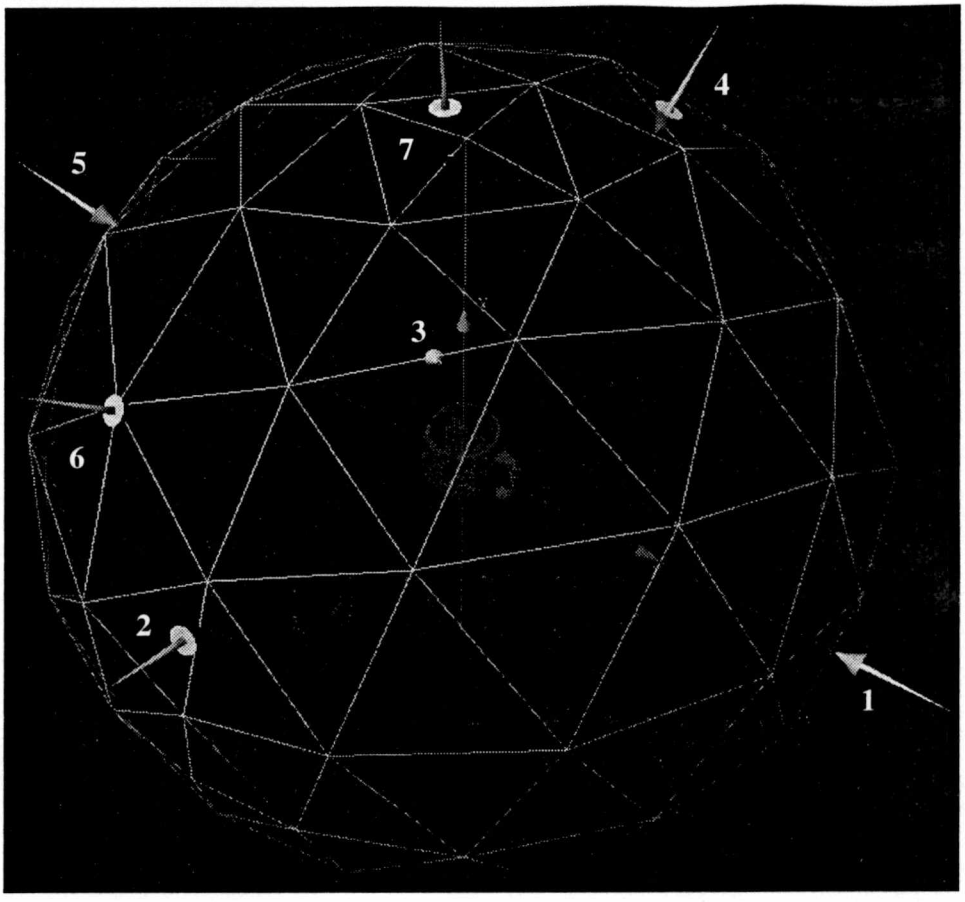


Figure 4.6: *The reconstructed 256×256×256 model of a coffee mug, with view positions shown as arrows pointing toward the object center.*

the visualization are the three coordinate axes, drawn as lines.

The matter of visualizing a triangulated isosurface of the binary model is more complex. As previously mentioned, several functions exists which take as input a 3-D model and output a surface representation of that model. One such function may be found in the IsosurfaceLat Explorer module. The module requires the input grid to be of Explorer Lattice geometry format. The Lattice is a C language structure containing the number of dimensions to the array and pointers to the dimensions themselves, the array data, and the coordinate type. The array may be an array of type byte, short, long, float, or double, and the coordinate type may be uniform, perimeter, or curvilinear. Our data is formatted into a uniformly-spaced array of raw bytes, with a range from 0 to 255.

Because the surface to be visualized will be an approximation of the actual object surface, and because we would like the surface to be smooth, we will pass the input grid through a smoothing filter before visualizing. Thus each voxel originally with a binary value of 0 (unoccupied) will correspond to an element in the byte array with a byte value of 0, and each voxel originally with a binary value of 1 (occupied) will correspond to an element in the byte array with a byte value of 255. The array will then be propagated through a Gaussian filter to smooth the transition from occupied to unoccupied.

The data will now be in a format appropriate for input into the IsosurfaceLat Explorer module. The module allows for interactive control of the threshold which determines the value at which an element is either inside or outside the object, and a flag may be set which will "smooth" the surface by shading the rendered polygons. We will use the intuitive threshold, half way between zero and 255 at 128, and will by default use the shading option. An isosurface rendering of the reconstructed coffee mug model is shown in Fig. 4.7.



Figure 4.7: *An isosurface rendering of the reconstructed coffee mug model.*

4.5 Summary

In this chapter, we have addressed issues relating to 3-D object representation. We have defined the formats of representation for our 3-D models. We have determined how we will translate the range data to three dimensions. Lastly, we have found the software applications we can use to visualize our occupancy grids.

In the following chapters we will describe the applications we have constructed based on the libraries and applications discussed in this chapter. Throughout the research into the best-next-view position, we will use the applications we have described here.

CHAPTER 5

DETERMINATION OF THE BEST-NEXT-VIEW

As stated in Chapter 1, our concern will be to position the range sensor such that an optimal amount of previously unrecorded information is assimilated into the knowledge base of the system with the next view acquisition. We will use this information to reconstruct a three-dimensional model as completely as possible, as quickly as possible. The product of a system which solves the best-next-view problem will be a minimum number of sensing operations which adequately reconstructs a 3-D model of an object or stationary environment.

According to the standard set by Kim in [35], we may be interpreted as addressing the issue of sensor placement in the context of MCSO (moving camera, stationary object), as illustrated in Fig. 5.1(a). This is equivalent to mapping the 3-D locations of points on a stationary object surface with a mobile range sensor. For practicality, we propose that we are also addressing the issue for application in the context of a stationary camera with a moving object (SCMO), as seen in Fig. 5.1(b). In this paradigm, the range sensor remains at a fixed location while acquiring range data from a 3-D object which is iteratively being rotated about its central axes. We do not restrict ourselves by saying that our system should be applied exclusively in one context or the other; this is because a best-next-view system which we produce may find application in either paradigm.

Having settled most of the fundamental issues regarding the system we wish to construct in previous chapters, we can move directly into deciding the next camera pose. We will note that because we are always to place the sensor at a fixed radius from some global origin, we are restricted to placing the camera only at locations on a sphere centered

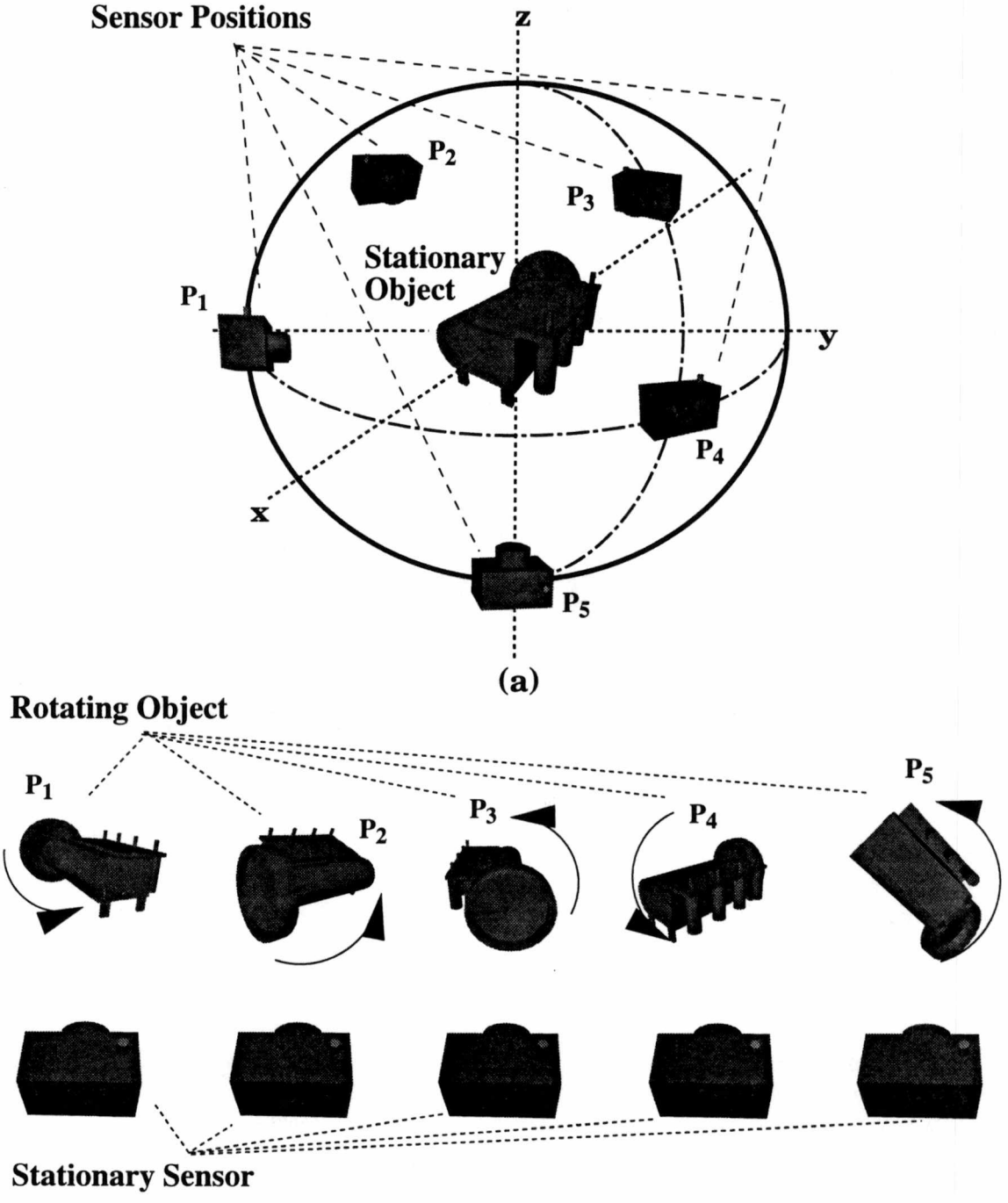


Figure 5.1: *The best-next-view system we will construct may be applied in the contexts of either (a) moving camera, stationary object (MCSO), or (b) stationary camera, moving object (SCMO).*

about the object or scene. We will also note that we decided to always orient the sensor in the direction of that origin. These restrictions do limit the acquisition capability of the sensor somewhat, but also guarantee the sensor will always acquire range data which describes the central portion of the scene. Because we will most generally be reconstructing models of objects entirely within the sensor's field of view, this condition helps to ensure that the sensor will acquire *relevant* 3-D data.

We would like to be able to input an existing 3-D model into the system and receive as output a position and orientation from which a range sensor would obtain the greatest quantity of previously unrevealed information. Because there is very rarely one method for solving a problem which will under every circumstance yield a more attractive result than any other method, a more complete system should be capable of solving a problem using one of a series of methods. This concept is well-illustrated by any naturally occurring intelligent system, which will generally make decisions regarding its environment based on contextual information, rather than on a single criteria. We will therefore hope to incorporate into the system multiple algorithms for determining the best-next-view position. A diversity of methods will increase the apparent intelligence of the developed system, and the conditional application of a method under pre-determined circumstances will increase the quality of, and hence the user's confidence in, any decisions made.

In the following sections, we will discuss four approaches we have developed for determining the best next sensor position. The first two sections introduce two simple approaches to sensor placement which consider only partial information in determining the next sensor position. Both approaches produce adequate results, and find justification in human intuition, but both also have inherent limitations. The latter two sections in this chapter introduce two "holistic" approaches to sensor placement, which take a more global view of the status of reconstruction. The justification for each technique will be stated as the technique is introduced, and the system that is the intelligent integration of all the methods will be described in Section 6.4.

5.1 Method 1: Sensor Positioning According to View History

An intuitive approach to range sensor placement would be to specifically consider positions farthest from previous sensing locations. The problem of implementing such a solution then lies in its formulation. We could assume an approach based on the Euclidean distance between vectors, so that the sensor is placed as far from every previous position as possible (with the constraint that the sensor remain at a fixed radius from the object center). In contrast, we could develop an approach which maximizes the difference of the spherical coordinate angles between a new view vector and previous view vectors. In theory, these approaches should yield an identical result.

Because we are already working with vectors in Cartesian coordinates, and because of the inherent disadvantages of mapping a point into the spherical coordinate system, we have chosen to implement a solution based on the Euclidean distance between view vectors. Before we consider a theoretical solution to this problem, we should discuss the complexity of such a solution. In order to find a vector which maximizes the distance between itself and other vectors, we must determine a local minimum of some distance function.

It can be shown that the total number of such minima will depend on the number of previous vectors. For instance, in two dimensions, only three points can be equidistant from each other on a circle; there are three points on the circle which will maximize the sum of the distances from that vector to previous points (see Fig. 5.2). Similarly, in three dimensions, up to four points can be equidistant on a sphere, and four points on the sphere maximize the sum of the distances from previous points. This corresponds to the solution of an fourth order polynomial, which will have up to four real roots.

We would like to find a point which maximizes the sum of the squared distances from that point to several fixed points on a sphere. Thus we want to find a local maximum of the function S, formulated by

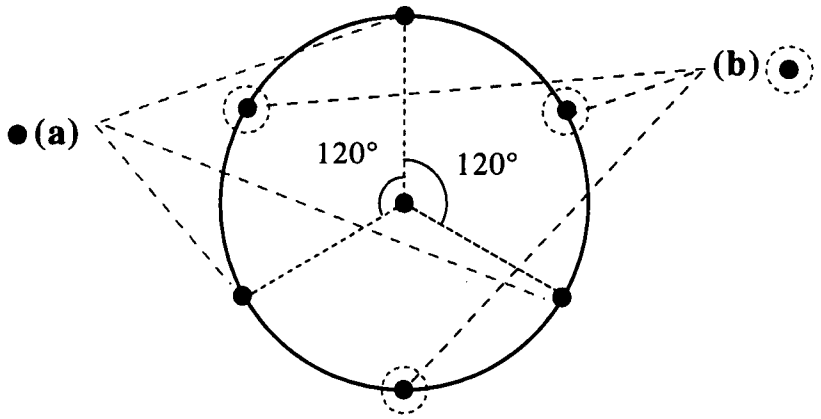


Figure 5.2: *(a) Three points on a circle which are equidistant from each other, and (b) the three points on the circle which maximize the sum of the distances from the point to previous points.*

$$S = \sum_i d_i^2 = \sum_i (x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2, \tag{5.1}$$

where d_i is the distance from the new point to the i^{th} fixed point, x, y, and z are the elements of the new point, and x_i, y_i, and z_i are the elements of the i^{th} fixed point. In order to calculate the maximum of the function S, we must differentiate it with respect to x, y and z separately and set the derivative equal to zero. If we proceed along this path, we will eventually be required to solve an n^{th} order polynomial, where n is the total number of views acquired. Because this procedure will be implemented on a computer, the practical step to take would be to develop an algorithm for finding one of these minima numerically, rather than theorizing on the solution of polynomials.

With this idea in mind, we will develop a numerical method for finding a point on a sphere which maximizes the sum of the distances from several fixed points on the sphere. A good initial guess for such a position $\mathbf{P_0}$ will be the mean of the previous fixed points $\mathbf{P_i}$ multiplied by −1:

$$\mathbf{P_0} = {}^{-1}/_{i} \sum_i \mathbf{P_i}. \tag{5.2}$$

If we assume a neural network-like approach to determining this position, we will iteratively increment the the initial value in a direction towards the correct solution. If this initial guess turns out to be the zero vector **0**=[0,0,0], or one of the previous fixed positions, we will randomize the elements of the initial vector. The iterative increment I should be proportional to the negative of each fixed position $\mathbf{P_i}$ and inversely proportional to the distance, d_i, from the current solution to that position:

$$I = \lambda (-\mathbf{P_i}) / d_i. \tag{5.3}$$

λ represents a gain factor to keep the increment from being either too large to saturate the system or too small for real-time processing.

Once implemented, this numerical algorithm actually converges rather quickly (dependent on the value of λ). It always finds at least a local maximum of the sum of the distances, and in general positions the sensor at a reasonable location. We will note, however, that this algorithm converges to the same result regardless of the object being reconstructed. This may be considered to be either an advantage or disadvantage, as we will see in the next chapter.

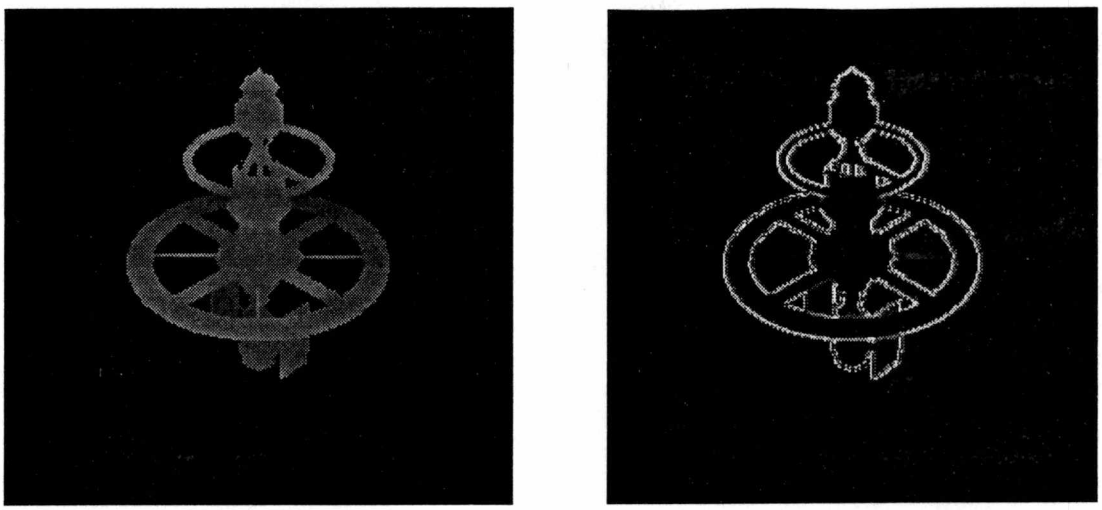
5.2 Method 2: Edge-Based Sensor Placement

The next approach we shall consider for positioning the range sensor takes into consideration only the information from the most recently acquired range image. Similar approaches were developed by Maver and Bajcsy in [39] and by Banta et al. in [5]. If we were to consider an individual range image without examining any 3-D model information inferred from previous data for the application of future sensor placement, we should make use of the information available to us as efficiently as possible. We would like for the sensor to acquire as much new data as possible with the next image acquisition. Thus we should examine the range image for evidence of occlusions, which will probably conceal a certain amount of surface information.

We can actually detect occlusions by measuring the level of discontinuity of the edges in the range image. An edge is simply a location where the transition between ranges of significantly different value is sudden; an edge in a range image will thus represent a surface nearly perpendicular to the current view direction. We can extract edge information from a range image by examining the difference between the intensities of adjacent pixels, i.e., by applying a proper curvature filter over the range data. An edge in a range image may be viewed as the visible portion of a surface nearly perpendicular to the present viewing direction. We can make a very general approximation of the normal of one of these "surfaces" by drawing a vector from the center of the image to an edge point.

The discrete gradient approach, approximated by the Prewitt operator, provides a suitable means for extracting edges from an image [29]. When this transformation in the form of a masked filter is applied to an area of discontinuity in a range image, the value of the gradient will be large relative to the values returned from neighboring continuous areas. This property will provide us with a simple method for detecting edges in the range image, and hence occlusions to the camera's view on the object's surface. Figure 5.3 illustrates a synthetic range image of a space station and the corresponding edge map derived using discrete gradient differentiation. We will note that the procedure appears to have extracted all major edges in the original image.

Most every range image will contain many occluding edges. If we assume that the information occluded by less significant edges can be considered to be negligible, we need only consider a limited number of the larger, more significant edges concealing greater quantities of unknown information. We will store the edge information in an edge map (as in Fig. 5.3(b)), in which we will locate the brightest intensity to determine the locations of the more significant edges. One can intuitively reason that the best direction to view all the surfaces around an edge is the direction formed by the combination of all the surface normal vectors around that edge (see Fig. 5.4). Below, we will mathematically prove this reasoning to be correct.



(a) **(b)**

Figure 5.3: *(a) A synthetic range image, and (b) the edge map corresponding to the image.*

Examining Fig. 5.4(a), we see two surfaces S_1 and S_2 with different surface normals $\mathbf{n_1}$ and $\mathbf{n_2}$. If we wish to view both surfaces from a single position, that position must lie in the region between the two surface planes. When viewing a single surface S, as shown in Fig. 5.4(b), the ability of the camera to distinguish between points on S will be proportional to $\cos(\theta)$, where θ is the angle between the view direction **d** and the surface normal **n** [9]. Thus the camera will have a maximum ability to discriminate between the positions of points on S when $(\delta \cos(\theta)/\delta\theta)=0$, which yields $\theta=0$. In other words, the best orientation from which to view the surface will be in the direction of the surface normal, **n**. Considering Fig. 5.4(c), the ability of the camera to simultaneously distinguish between points on surfaces S_1 and S_2 will be proportional to $(\cos(\theta_1) + \cos(\theta_2))$. To calculate the orientation of the best view for both surfaces, we differentiate:

$$\delta/_{\delta\theta_1} [\cos(\theta_1) + \cos(\theta_2)] = 0. \qquad \textbf{(5.4)}$$

Since we are differentiating with respect to θ_1, we substitute $\theta_2=\phi-\theta_1$, and $\cos(\phi-\theta_1)=\cos(\theta_1-\phi)$, so

$$\delta/_{\delta\theta_1} [\cos(\theta_1) + \cos(\theta_1-\phi)] = 0, \qquad \textbf{(5.5)}$$

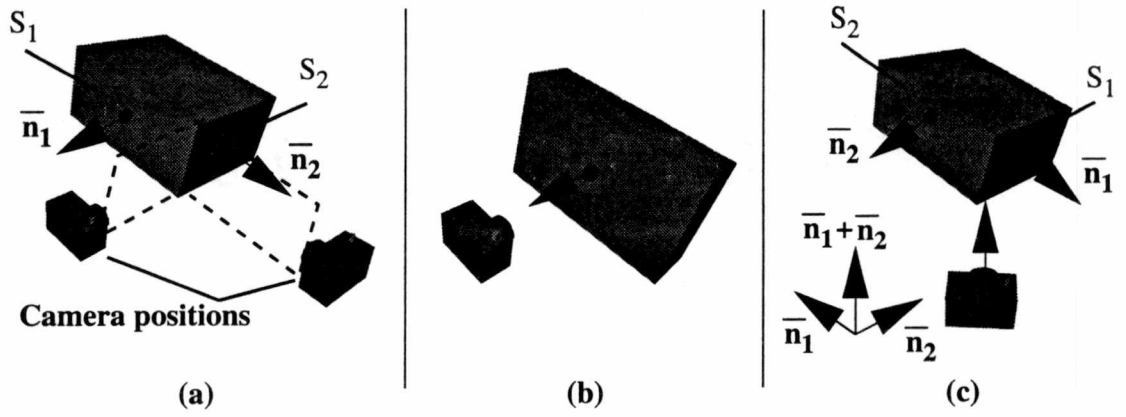


Figure 5.4: *The best view position for two surfaces: (a) In order to view two surfaces S_1 and S_2 with surface normals $\mathbf{n_1}$ and $\mathbf{n_2}$, the camera must be in the specified region between the surface planes. (b) The best position to view a surface S is in the surface normal direction $\mathbf{n}$ over the center of the surface. (c) The best position to view two surfaces S_1 and S_2 is thus in the direction that is the combination, $(\mathbf{n_1}+\mathbf{n_2})$, of the surface normal directions.*

which simply yields $\phi = 2\theta_1$, and since $\phi = \theta_1+\theta_2 = 2\theta_1$, we deduce that $\theta_1=\theta_2$.

Because $\mathbf{n_1}$ and $\mathbf{n_2}$ are both unit vectors, the direction at equal angles from $\mathbf{n_1}$ and $\mathbf{n_2}$ is defined by their sum, $(\mathbf{n_1}+ \mathbf{n_2})$. Similarly, in a more complicated case such as that in which an edge is formed by the joining of *m* surfaces, where *m* is an integer greater than one, the orientation which will provide the best view of all surfaces is [59]:

$$\mathbf{d} = \sum_i \mathbf{n_i}, \tag{5.6}$$

where $\mathbf{n_i}$ is the unit normal vector of the i^{th} surface.

Once we have obtained the edge map of the range image, we will determine the locations of the pixels of the brightest intensity in the edge map. Following the theory discussed above, we will then average the locations of the pixels which have that intensity. The vector pointing from the center of the image to the brightest pixel average will be the direction we shall use to define $\mathbf{n_{os}}= (x_{os},y_{os})$, the normal of the surface which is

orthogonal to the camera's view (see Fig. 5.5). We will transform this vector to three dimensions to define the best-next-view vector.

The transformation from two to three dimensions will require knowledge of the image orientation and acquisition point. The sensor position relative to the object will define the vector $\mathbf{v_{prev}}$, which points *into* the image. Additionally, the image will have been acquired under the condition that the image up direction $\mathbf{d_{iu}}$ is as close as possible to an absolute up direction **u**. In other words, the vector **u** will be made up only of components $\mathbf{u_{v_{prev}}}$ in the direction of $\mathbf{v_{prev}}$ and $\mathbf{u_{d_{iu}}}$ in the direction of $\mathbf{d_{iu}}$ (see Fig. 5.6). From simple vector algebra, we know that the dot product D of two vectors **a** and **b** is represented as

$$D = \mathbf{a} \cdot \mathbf{b}, \tag{5.7}$$

With the assumption that any vector may be decomposed into components of any two orthonormal vectors, and that the component $\mathbf{x_a}$ of a vector **x** in the direction of a unit vector **a** is given by

$$\mathbf{x_a} = (\mathbf{a} \cdot \mathbf{x})\,\mathbf{a}, \tag{5.8}$$

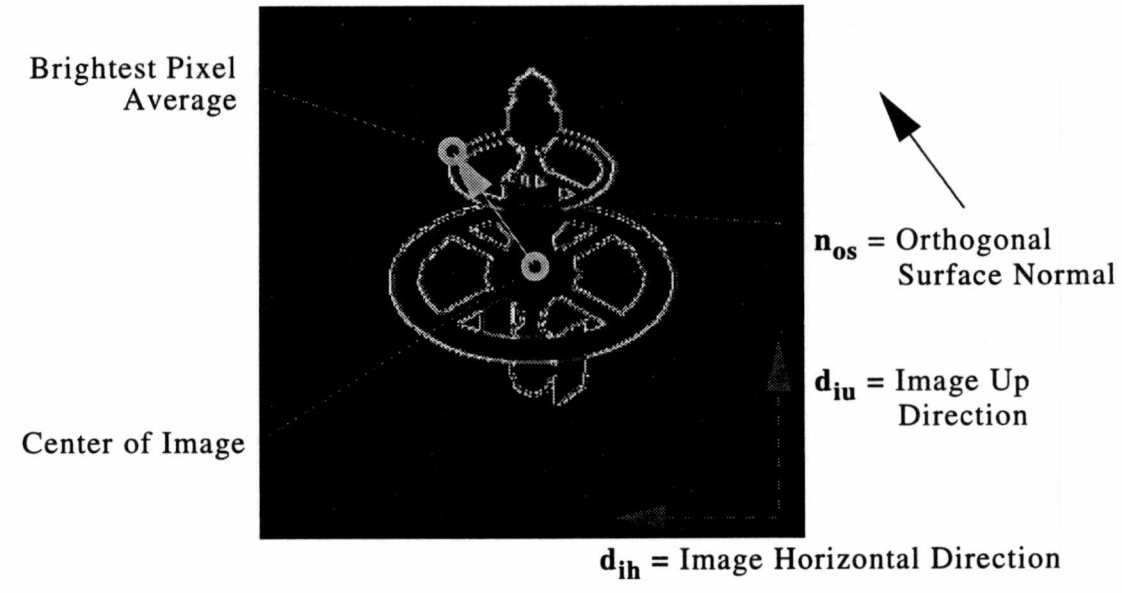


Figure 5.5: *The vector pointing from the center of the image to the brightest pixel average will be the direction we shall use to define the orthogonal surface normal.*

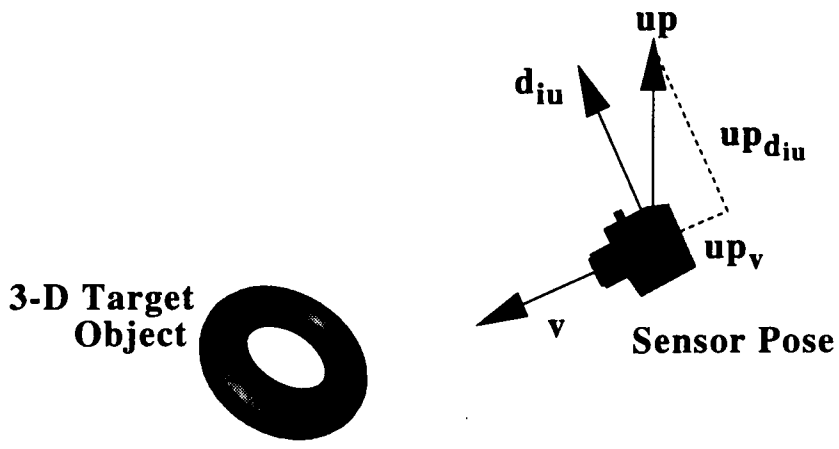


Figure 5.6: *The image up direction d_{iu} may be calculated from the view vector v and the up direction* ***up***.

we may state that a vector **x** can be decomposed into components of orthonormal vectors **a** and **b**:

$$\mathbf{x} = \mathbf{x_a} + \mathbf{x_b} = (\mathbf{a} \cdot \mathbf{x})\,\mathbf{a} + (\mathbf{b} \cdot \mathbf{x})\,\mathbf{b}. \tag{5.9}$$

If we substitute into this equation **u** for **x**, $\mathbf{u_{d_{iu}}}$ for $\mathbf{x_a}$, and $\mathbf{v_{prev}}$ for **b**, we have:

$$\mathbf{u} = \mathbf{u_{d_{iu}}} + (\mathbf{v_{prev}} \cdot \mathbf{u})\,\mathbf{v_{prev}}. \tag{5.10}$$

This equation is easily solved for $\mathbf{u_{d_{iu}}}$:

$$\mathbf{u_{d_{iu}}} = \mathbf{u} - (\mathbf{v_{prev}} \cdot \mathbf{u})\,\mathbf{v_{prev}}. \tag{5.11}$$

Now that we have the component of **up** in the direction of $\mathbf{d_{iu}}$, we simply normalize the component to find $\mathbf{d_{iu}}$:

$$\mathbf{d_{iu}} = \mathbf{u_{d_{iu}}} / |\,\mathbf{u_{d_{iu}}}|. \tag{5.12}$$

Once we have calculated $\mathbf{d_{iu}}$, we need only to calculate the image horizontal direction $\mathbf{d_{ih}}$. Because it is the direction that is orthogonal to both $\mathbf{v_{prev}}$ and $\mathbf{d_{iu}}$, we may calculate the value of $\mathbf{d_{ih}}$ by taking the cross product of $\mathbf{d_{iu}}$ and $\mathbf{v_{prev}}$:

$$\mathbf{d_{ih}} = \mathbf{d_{iu}} \times \mathbf{v_{prev}}. \tag{5.13}$$

We now have the two orthonormal vectors, $\mathbf{d_{ih}}$ and $\mathbf{d_{iu}}$, which define the image plane. We have already found the normal of the surface orthogonal to the camera's view $\mathbf{n_{os}}=(x_{os},y_{os})$ in image coordinates, using the average of the brightest pixels in the edge map. We now translate the two-dimensional normal vector $\mathbf{n_{os}}$ to three dimensions to define the next view vector $\mathbf{v_{next}}$:

$$\mathbf{v_{next}} = x_{os}\mathbf{d_{ih}} + y_{os}\mathbf{d_{iu}}. \tag{5.14}$$

We can normalize $\mathbf{v_{next}}$ to whatever magnitude is appropriate for the view radius. Because $\mathbf{d_{ih}}$ and $\mathbf{d_{iu}}$ are both orthogonal to the previous view vector $\mathbf{v_{prev}}$, the next view vector $\mathbf{v_{next}}$ as calculated using this method will always be orthogonal to the most recent view vector.

The complete procedure as stated above for determining the best next view position can be summarized as follows:

1. Calculate the edge map of the most recently acquired range image by applying a gradient filter [29].
2. Find the average location of the brightest pixel value in the edge map $\mathbf{n_{os}}$ relative to the center of the image.
3. Calculate the horizontal and vertical directions of the image $\mathbf{d_{ih}}$ and $\mathbf{d_{iu}}$ based on Eqs. 5.11-13.
4. Translate the two-dimensional $\mathbf{n_{os}}$ to three dimensions using Eq. 5.14 to find the next view vector $\mathbf{v_{next}}$.

5.3 Method 3: Targeting the Occluded Voxel Face Centroid

In this section and the following section, we will discuss two techniques for determining future sensor positions which take into consideration the entirety of the previously acquired model information, and may be considered "holistic" approaches to sensor placement. The methods discussed in the previous two sections considered only partial information in determining the next sensor pose.

As we explained in Section 4.3.4, the models we will reconstruct from range data will be composed of surface information obtained directly from range data *and* the information inferred from the range data which was occluded from the range camera's view (refer to Fig. 4.5). The combined information assimilated from each view position may then be employed to determine the next sensing operation which will acquire the greatest quantity of new information. We might first consider an approach which will orient the range sensor on the greatest volume of occluded voxels. This idea has several inherent flaws. The most obvious is that portions of the occluded model will be exposed and other portions will be covered by the recovered surface model. Thus, orienting the sensor on a volume could place the sensor at a seemingly random position. Also, even when the model has been almost fully reconstructed, there will be occluded information inside of the surface model; searching for an occluded volume will probably place the sensor where it has already been. Lastly, the question will arise as to how to position the sensor given the occluded volume: because the volume does not have a specific orientation, there is no way to predict the direction from which it will be visible.

The solution to all of these problems is to consider primarily the *surface* of the reconstructed model. By using the exposed surface of the occluded model to define the next sensor pose, the system can orient the sensor exactly on the portion of the model where it is least accurate and assimilate new information into that portion of the model. The "surface" of the discrete model will be comprised of a mesh of exposed voxel faces. An exposed voxel face will be found at any location where an occupied voxel in the 3-D grid is adjacent to at least one unoccupied voxel. The directional orientation (normal) of a surface patch can be determined from the summed orientations of the voxel faces comprising the patch, and this orientation can be used to determine a position from which the patch will be visible.

Obviously we would like to view the greatest area of the occluded surface possible. One way to accomplish this is to orient the sensor on the centroid of the surface. One can

reason that this approach will perform well when confronted with one continuous occluded surface, as shown in Fig. 5.7(a). The centroid of the surface is surrounded by information which will prove useful in reconstructing a 3-D model. However, in the case of Fig. 5.7(b), where the occluded model is exposed in several different surface patches and one cannot predict where the centroid of the surface will be, the centroid approach might position the sensor at a seemingly ambiguous location. This leads us to conclude that a centroid approach would perform well during the initial view acquisitions, but the performance would degrade as reconstruction progressed. We will take these issues into consideration when integrating the methods into an intelligent autonomous best-next-view system.

The centroid of the surface may be found using one of several techniques. The most obvious of these is to find the mean voxel face location on the occluded surface. The mean $\mathbf{m_{3\text{-}D}}$ of a 3-D data set S will simply be the vector that is the sum of all sample vectors $\mathbf{S_i}$ divided by the number of samples n:

$$\mathbf{m_{3\text{-}D}} = (1/n) \sum_i \mathbf{S_i}. \tag{5.16}$$

There are several alternative approaches to finding the centroid of a surface. In contrast to the averaging approach discussed above, we consider a technique which calculates the *median* of the exposed voxel faces in the occluded surface to find the centroid of the surface. The primary advantage of a median approach is that it would not be as susceptible to statistical outliers. As long as the number of samples i is reasonably large, the median will be a good estimate of the sample centroid. A surface median technique will otherwise be very similar to a voxel face average technique. One similarity is that, due to the multi-dimensionality of the surface, the median voxel face location might not actually be on the occluded surface. This is because a 3-D median will be a vector composed of elements which are the independent medians of x, y, and z sampling. In other words, if we have a 3-D data set **S** consisting of i samples, we will find the median corresponding to each coordinate axis by ordering the samples with respect to that axis and finding the location of

the $(i/2)^{th}$ sample. The median vector $\mathbf{M_{3\text{-}D}}$ will thus be given by:

$$\mathbf{M_{3\text{-}D}} = [x_{i/2}, y_{i/2}, z_{i/2}]. \tag{5.18}$$

We must also keep in mind that if the occluded surface is well-distributed about the center of the model or one of the coordinate axes, the centroid of the surface will be nearer to the object center than the object surface (see Fig. 5.8). If the portions of the surface that are well-distributed about one of the coordinate axes represent a significant portion of the total surface, then this representation of the centroid would not yield a good target point for the range sensor. Thus we should somehow consider the surface distribution about the object center in our selection of a surface centroid.

A simple way to do this would be to make note of the exposed positive and negative **x**, **y**, and **z** faces separately and select an orientation so that the positive and negative do not cancel out one another's contribution to the surface normal. Of course, if the number of positive faces does not compare well with the number of negative faces (i.e., if the ratio between them is less than 0.9 or greater than 1.1) then we simply use the combination of the two. Thus we will make note of the numbers of exposed ±**x**, ±**y**, and ±**z** faces (n_{mx}, n_{px},

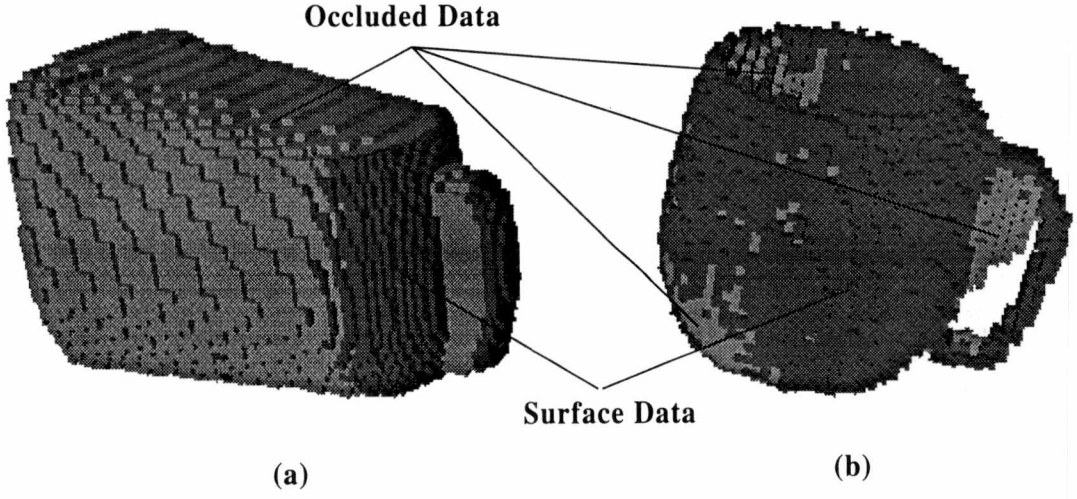


Figure 5.7: *(a) A model in which the exposed surface of the occluded model is fairly continuous, and (b) a model in which the exposed occluded surface is composed of several disjoint surface patches.*

n_{my}, n_{py}, n_{mz}, and n_{pz}) in the occluded model and the location in space of each face, and calculate the three ratios R_x, R_y, and R_z of exposed positive faces to exposed negative faces:

$$\begin{aligned} R_x &= n_{px}/n_{mx} \\ R_y &= n_{py}/n_{my} \\ R_z &= n_{pz}/n_{mz}. \end{aligned} \tag{5.19}$$

The centroid voxel face locations $\mathbf{C_{px}}$, $\mathbf{C_{mx}}$, $\mathbf{C_{py}}$, $\mathbf{C_{my}}$, $\mathbf{C_{pz}}$, and $\mathbf{C_{mz}}$ will then be calculated for $\pm\mathbf{x}$, $\pm\mathbf{y}$, and $\pm\mathbf{z}$ according to Eq. 5.16 or Eq. 5.18. If the ratio of exposed positive faces to exposed negative faces is between 0.9 and 1.1, we define the coordinate centroids $\mathbf{C_x}$, $\mathbf{C_y}$, and $\mathbf{C_z}$ as the mean or median accounting for the larger number of exposed faces; otherwise, we use the overall centroid voxel face location to define the coordinate centroid:

$$\mathbf{C_x} = \begin{cases} \mathbf{C_{px}} & R_x > 1.11 \\ \mathbf{C_{mx}} & R_x < 0.90 \\ (n_{px}\mathbf{C_{px}}+n_{mx}\mathbf{C_{mx}})/(n_{px}+n_{mx}) & (R_x \geq 0.90) \text{ AND } (R_x \leq 1.11) \end{cases} \tag{5.20(a)}$$

$$\mathbf{C_y} = \begin{cases} \mathbf{C_{py}} & R_y > 1.11 \\ \mathbf{C_{my}} & R_y < 0.90 \\ (n_{py}\mathbf{C_{py}}+n_{my}\mathbf{C_{my}})/(n_{py}+n_{my}) & (R_y \geq 0.90) \text{ AND } (R_y \leq 1.11) \end{cases} \tag{5.20b)}$$

$$\mathbf{C_z} = \begin{cases} \mathbf{C_{pz}} & R_z > 1.11 \\ \mathbf{C_{mz}} & R_z < 0.90 \\ (n_{pz}\mathbf{C_{pz}}+n_{mz}\mathbf{C_{mz}})/(n_{pz}+n_{mz}) & (R_z \geq 0.90) \text{ AND } (R_z \leq 1.11) \end{cases} \tag{5.20(c)}$$

We must then make note of the number of voxel faces n_x, n_y, and n_z each centroid represents, and compute the weighted average of the three coordinate centroids to determine the overall surface centroid **C**:

$$\mathbf{C} = (n_x\mathbf{C_x} + n_y\mathbf{C_y} + n_z\mathbf{C_z})/(n_x + n_y + n_z). \tag{5.21}$$

The nearest voxel face $\mathbf{F_{min}}$ to the centroid **C** is then easily found by iteratively comparing distances between the centroid and all exposed voxel faces $\mathbf{F_i}$:

$$\| \mathbf{F_{min}} - \mathbf{C} \| = \min \| \mathbf{F_i} - \mathbf{C} \|. \quad (5.22)$$

Once we have located the centroid of the object surface, we will trace a ray in the direction of the surface normal from the voxel face nearest to the centroid to the view sphere to determine the next sensor pose. The elements of the surface normal $\mathbf{N_s}$ can simply be calculated as the number of exposed voxel faces in the corresponding directions:

$$\mathbf{N_s} = [n_{px} - n_{mx} , n_{py} - n_{my} , n_{pz} - n_{mz}]. \quad (5.23)$$

Because we will have selected either the positive, negative, or combination of positive and negative exposures, the vector we shall use for the surface normal will be defined by only the number of exposed voxel faces in the appropriate directions.

5.4 Method 4: Clustering the Occluded Surface Data

Every approach for sensor placement we have proposed up to this point will perform well during initial view acquisitions, but none of them localize the search for occluded surface information. This localization will be necessary during later view acquisitions when the

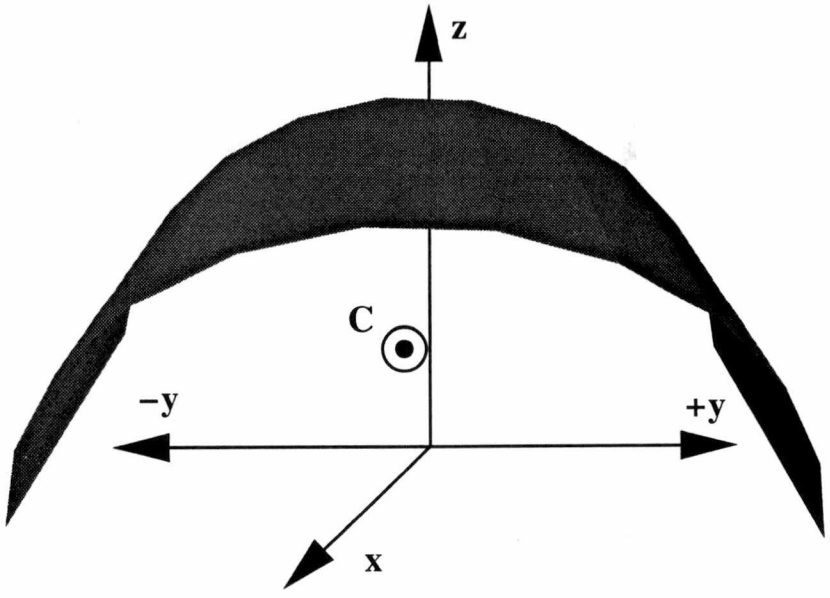


Figure 5.8: *Because the surface is evenly distributed about the y axis, the centroid C of the surface is located at y=0.*

occluded surface is broken up into several disjoint surface patches (see Fig. 5.8(b)). The obvious conclusion is that we should develop a sensor positioning technique for use during later view acquisitions which will isolate some of the more significant surface patches and position the sensor such that the information necessary to reconstruct that portion of the model could be acquired. The issue which remains to be settled, then, is the method we will use to isolate the significant surface patches.

Essentially, the function we would like to perform is the clustering of 3-D data. The system we desire should take as input a list of exposed voxel faces in the occluded model and cluster the faces into an unspecified number of tightly-packed patches. There are a great number of existing methods for clustering data, but most found in our research performed only *supervised* clustering (i.e., the number of clusters to be found was pre-specified). Because we certainly will not know how many surface patches the occluded model will present, we must develop an approach for *unsupervised* clustering (i.e., the number of clusters to be found is unknown).

The idea of clustering data involves somehow measuring the similarity between vectors within a data set and placing the vectors most similar to one another together in clusters. The most intuitive measure of similarity is that of Euclidean distance (e.g., the most similar vectors will be those that are closest together). The clustering approach we have chosen to develop is a variation of the well-known *maximin-distance algorithm* [29]. This clustering approach tends to find clusters which are closely-packed, and thus is well-suited for separating disjoint surface patches. The algorithm begins with each voxel face initially belonging to the same cluster, and proceeds as follows:

Step 1: Set each cluster's sample mean as its prototype.
Step 2: Compute the distance from each prototype to each voxel face.
Step 3: Make each face a member of the nearest prototype's cluster.
Step 4: Find the face $\mathbf{f_0}$ farthest from its prototype.
Step 5: Find the mean distance between cluster prototypes d_{mean}.

Step 6: If the face $\mathbf{f_0}$ is half as far from its prototype as the mean distance between prototypes d_{mean}, make it the only member of a new cluster and go to Step 1 (else quit).

Steps one through six are repeated until the farthest voxel face from its cluster prototype is less than half as far from its prototype as the mean distance between prototypes.

The next step we will perform will be to find the cluster containing the largest number of faces. It is this cluster of exposed voxel faces on which we will orient the range sensor. During the above clustering procedure, we will have already computed the center of the cluster (and therefore the centroid of the surface) and all the distances from each voxel face to the cluster prototype. The target point on which we should orient the sensor will simply be the voxel face belonging to the largest cluster for which the distance from it to the cluster prototype is minimum. Because the cluster will be a local group of voxel faces, we need not concern ourselves with the distribution of the voxel faces about the centroid of the surface, as with the mean and median approaches discussed previously. The **x**, **y**, and **z** coordinate elements of the surface normal will again be the number of exposed voxel faces in each corresponding direction, as formulated in Eq. 5.23.

CHAPTER 6

SYSTEM OPTIMIZATION, INTEGRATION, AND IMPLEMENTATION

In Sections 6.1 through 6.3, we will consider three separate issues concerning a best-next-view system. In Section 6.1, we will consider the situation in which the best-next-view system produces a view position from which the area of interest is not visible to the sensor. In Section 6.2, we will discuss the situation in which the system places the sensor unnecessarily close to a previous sensing location. Section 6.1 will define the criteria which will determine when the system has adequately reconstructed a model. Because these situations are somewhat complex, a certain amount of attention and detail must be given each topic. In Section 6.4, we will use our knowledge of the approaches we have developed to assimilate them into an intelligent autonomous best-next-view system. The complete system can then take as input images from the image acquisition function described in Chapter 3, and the range data will be registered using the registration algorithm described in Chapter 4. In Section 6.5, we will describe the software we have compiled which is the implementation of the best-next-view system.

6.1 Providing the Sensor with an Unobstructed View of a Feature

Frequently, when determining future sensor locations, one of the fundamental issues which will arise is the determination of a location from which a particular area of interest is visible. This issue comes to question particularly when we are attempting to reconstruct a model of an object with self-occluding concavities. An example of such an object is illustrated in Fig. 6.1. If we consider a situation in which the region of interest lies on the

interior of the concavity, it is highly probable that a best-next-view system will return a sensor position from which a view of the feature is occluded by the opposite edge of the concavity (depending on how the sensor position is calculated) [47]. This is because such systems (including our own) commonly rely on the normal of the surface of interest in order to place the sensor, and the normal inside a concavity will generally intersect the opposite edge of the concavity.

For those sensor placement approaches which determine view positions based on the state of the current model (e.g., Methods 3 and 4 of Sections 5.3 and 5.4, respectively), it is possible to assure that the target feature is actually visible from the product viewpoint. Because a portion of the model will already be reconstructed, we can use the shape of the reconstructed model to determine if the view of a feature is occluded from a specific point. This can be done by tracing a ray from the feature of interest to the surface of the view sphere in the specified direction, and checking for occupied voxels anywhere on the ray. If any voxel intersected by the ray is occupied, the view of the feature of interest will be at least partially occluded.

To find a position from which the feature is visible, we will start with a prespecified target point on the object surface and a view direction. If a ray oriented in the view

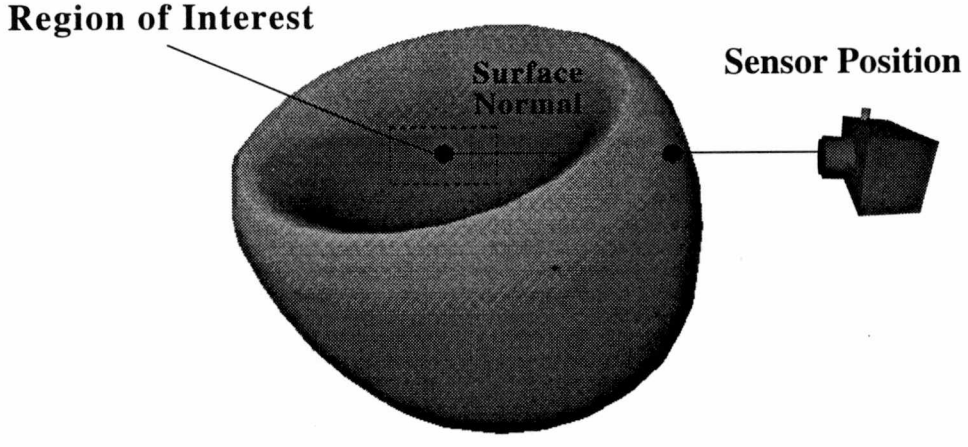


Figure 6.1: *An object which contains a self-occluding concavity.*

direction from the target point intersects any occupied voxels in the reconstructed object model, a view of the feature will be occluded. The solution to this problem is to slightly perturb the angle of the view direction. If we transform the Cartesian vector (x,y,z) which represents the view direction to spherical coordinates, we will have a vector (r,θ,ϕ), comprised of a radius and two angles. To find the closest view direction for which the feature is not occluded, we can iterate θ and ϕ through the range of angles within $\pm 90°$ ($\pm\pi/2$ radians), checking the ray corresponding to each potential direction for occlusions. This procedure thus considers only the half-sphere of points from which the surface might be visible (see Fig. 6.2).

The algorithm we have implemented in software takes as input an occupancy grid, a target point, and a view direction. Holding θ constant, we first iterate ϕ through all angles within 90° of the original view direction, in increments of one degree. The algorithm checks the status of all voxels on a line in the calculated direction radiated from the target point; if an occupied voxel is encountered, the next angle is checked. If no unoccluded directions are found, θ is incremented by one degree, and ϕ is iterated through all angles within 90°. In this fashion, we iterate θ through all angles within 90°. If no unoccluded direction is found, de-occlusion fails. This will not happen often, because the feature of interest is exposed to begin with, so it will probably be visible from some direction.

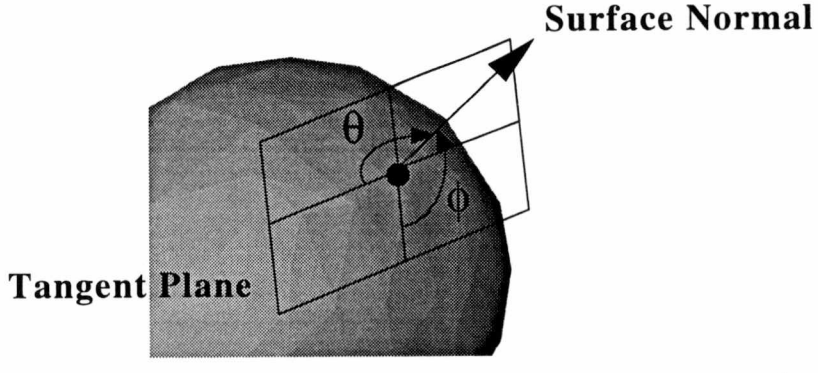


Figure 6.2: *A surface normal vector may be rotated about its two angular spherical coordinate axes and consequently ray-traced to determine an unoccluded view direction.*

Now, when implementing Methods 3 and 4, we can take the product target point on the occluded object surface and the normal of the surface of interest as the view direction, and input them into the function described above to assure that the resulting sensor position has a clear and unobstructed view of the feature of interest. In this manner, we can use the above described function to insure that all view positions returned by Methods 3 and 4 are truly valid.

6.2 Separating Closely-Spaced Views

In an automated sensor placement experiment, it is possible that the positioning system will place the sensor relatively close to a previous sensing position. Although this condition is undesirable, it is understandable because the automated system will only be as intelligent as we choose to make it. An example situation is that in which the de-occlusion of a view direction (see Section 6.1) shifts the returned sensor pose toward a previous sensor pose. Keeping this situation in mind, we should guard against the case of closely-spaced views.

Figure 6.3 illustrates a situation in which the sensor-placement system has determined a sensor position which is unnecessarily close to a previous sensor position. The intuitive solution to this problem will be to "push" the new view position away from the previous position, and this is the solution we have implemented in software. If we consider each view position to be a vector, we would like to set a minimum angle between the new vector

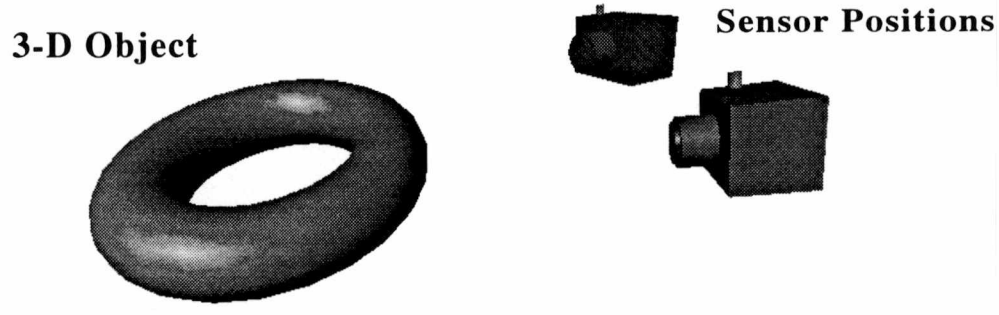


Figure 6.3: *A situation in which the sensor-placement system has determined a sensor position which is unnecessarily close to a previous sensor position.*

and each previous vector. The angle, α, between two vectors may be determined by calculating the arc-cosine of the dot product of the two vectors:

$$\alpha = \cos^{-1}(\mathbf{a} \cdot \mathbf{b}). \tag{6.2}$$

Since our goal is to separate vectors of the same radius by a pre-specified angle, we would like to keep the shifted vector in the plane that is already common to the two vectors (see Fig. 6.4).

On this basis, if we assume that our system has returned a view position represented by the vector $\mathbf{p_{new}}$, and the position is close to a previous view position represented by the vector $\mathbf{P_{prev}}$, we can separate $\mathbf{p_{new}}$ into components of the vector $\mathbf{P_{prev}}$ (normalized) and an orthonormal vector $\mathbf{Q_{prev}}$ (see Fig. 6.5(a)). We can determine the elements of $\mathbf{Q_{prev}}$ by substituting $\mathbf{P_{prev}}$ into Eq. 5.9 for **a** and $\mathbf{p_{new}}$ for **x** and solving for $\mathbf{Q_{prev}}$ as **b**:

$$\mathbf{Q_{prev}} = \mathbf{p_{new}} - (\mathbf{p_{new}} \cdot \mathbf{P_{prev}})\,\mathbf{P_{prev}}. \tag{6.5}$$

$\mathbf{Q_{prev}}$ must then be normalized. Now, if the angle α between $\mathbf{p_{new}}$ and $\mathbf{P_{prev}}$, given by $\cos^{-1}(\mathbf{p_{new}} \cdot \mathbf{P_{prev}})$, is less than the pre-specified threshold angle β, we can determine a new

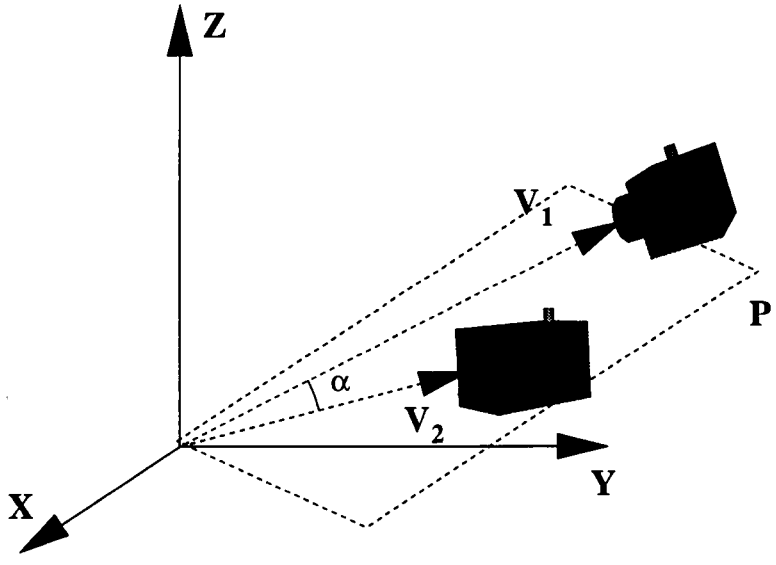


Figure 6.4: *Any two view positions originating from the same point, represented by vectors V_1 and V_2, will share a common plane P.*

location, $\mathbf{P_{new}}$, which is essentially a shifted $\mathbf{p_{new}}$:

$$\mathbf{P_{new}} = \mathbf{P_{prev}} \cos \beta + \mathbf{Q_{prev}} \sin \beta. \tag{6.6}$$

So that the magnitude of the vector is correct, we then multiply the vector by the previous magnitude, $|\mathbf{p_{new}}|$. An illustration of a shifted view position is shown in Fig. 6.5(b).

We now have a function which will assure that the next view position is not unnecessarily close to any previous sensing positions. The situation might exist, however, when a user is not be concerned with the next sensor position being relatively close to previous positions. For this reason, we will make the view separation function optional, and it may be engaged by the user whenever it is determined that an appropriate situation for the function's use exists.

6.3 Criteria for Reconstruction Termination

Since the system we wish to compile is to *manually* reconstruct a 3-D model, it should be capable of sensing when scene information is no longer necessary for reconstruction. Therefore we should determine the criteria which the system should examine before

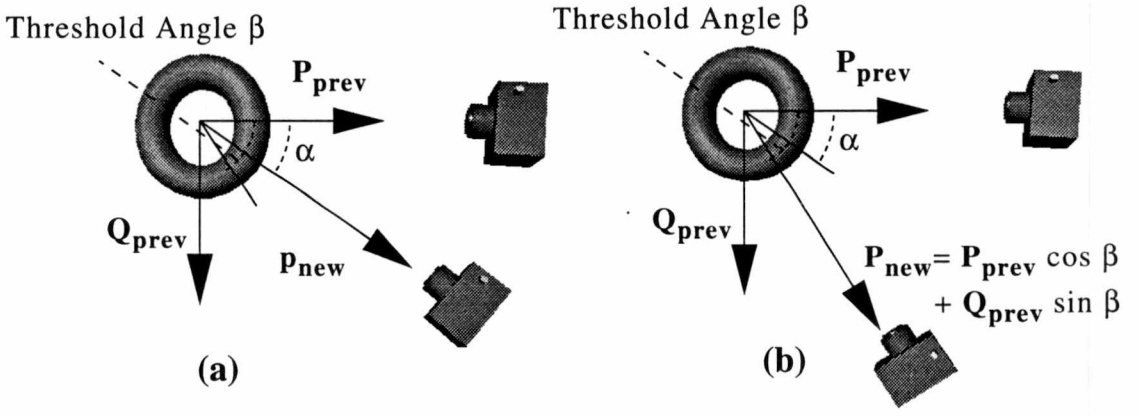


Figure 6.5: *(a) Because the angle α between the previous view position $\mathbf{P_{prev}}$ and the new view position $\mathbf{p_{new}}$ is less than the threshold angle β, (b) a new value is calculated for $\mathbf{P_{new}}$ based on a normalized $\mathbf{P_{prev}}$ and an orthonormal vector $\mathbf{Q_{prev}}$.*

self-terminating.

Because we are using the exposed voxel faces to visualize the models, and we will also use the exposed voxel face information for some of our view-determining techniques, it would be practical to also use this information to determine the algorithm termination criteria. The conditions which we would like the completely reconstructed model to meet should be intuitive. First of all, we would like for the reconstructed surface area to be substantially larger than the occluded surface area. This will be true if the model has been sufficiently reconstructed. However, we should obviously continue reconstruction if a relatively large amount of new data is being acquired with each view. Thus we should take into consideration the surface area of the model after the acquisition of previous views.

The condition which we should examine to determine if new information has been acquired may be set by either the surface model or the occluded model. We could examine the difference in size of the surface model after the acquisition of the present information and the previous view: if the surface model has increased in size by less than 10% with the assimilation of the most recent range data set, we could declare the model to be sufficiently reconstructed. However, we could say the same if the surface area of the occluded model decreased by less than 10%. The heuristic which we shall use will be to declare the present status of reconstruction to be sufficient if either is true. The system will thus terminate reconstruction if:

1. The ratio of surface faces to occluded faces is large (ratio>20) AND
2. The change in surface face count is small (change<10%)
   OR
   The change in occluded face count is small (change<10%).

Thus our system will automatically terminate reconstruction when the reconstructed model contains mostly surface information and it has ceased to change by relatively large amounts with the acquisition of each view.

6.4 An Intelligent Collage of Best-Next-View Methods

A well-constructed best-next-view system will attempt to emulate the reasoning used by a person during the evaluation of a 3-D object. Unfortunately, the faculties available to a human will not be available to the system. The system will be required to adequately reconstruct a 3-D model of an object given a discrete number of range views, whereas a person could survey the object from a continuous trajectory.

While the criteria for algorithm termination was set in Section 6.3, we have as yet to completely automate the system; in order for a best-next-view system to be simultaneously intelligent and completely autonomous, it must be capable of selecting the *best* method for determining the next view position at the current stage of reconstruction. In the previous sections, several such methods were introduced and discussed, including the strengths and weaknesses of each method. Some approaches would perform more optimally during the initial stages of reconstruction, while others were deemed appropriate for examining a more diversely distributed model. The edge map approach will always position the sensor orthogonally to the most recent sensor position, but the location of the new position relative to previous positions is arbitrary. For this reason, the edge map approach might not function optimally in the later stages of reconstruction. Similarly, the cluster technique attempts to divide the surface of the occluded model into several well-packed, disjoint surface patches, which might not yet exist in the initial stages of reconstruction.

The method used to determine the next view position at each iterative step of reconstruction will be determined using this knowledge. The position from which the first images are acquired is arbitrary; it will either be set to a default position or it may be specified. So that the second view vector will always be orthogonal on the view sphere to the first, the second view position shall be determined using the edge-based technique of Section 5.2. This method will be optimally used in the early stages of reconstruction, as we know that the method does not consider any *a priori* knowledge when positioning the

sensor, and hence might orient the sensor on an occlusion that has already been reconstructed. Also, due to the orthogonality property of the method, a surplus of new information should always be acquired with the second view.

For the intermediate third and fourth view positions, we will use the semi-intelligent occluded surface centroid approach. When reconstruction is in this stage, it is probable that the surface of the occluded model will be fairly continuous, so the overall centroid of the occluded surface should yield the sensor a good deal of previously occluded information. The surface centroid may be calculated as either the mean voxel face location (as in Eq. 5.16), or the median voxel face location (as in Eq. 5.18). While a centroid calculated as the surface median will generally be nearly identical to a centroid calculated as the surface mean, the median approach will not be as susceptible to statistical outliers in the surface data. Thus we shall calculate the surface centroid as the median of the data set.

Theoretically, the most optimal method for use during the latter stages of reconstruction will be the clustering approach of Section 5.4. However, the possibility does exist (though it is unlikely) that the approach, when applied repeatedly, will get caught in a local minimum and orient the sensor on uninformative regions of the object surface. An intelligent system would optimally safeguard against this possibility. One possible way to do this would be to monitor the rate at which reconstruction is proceeding with the acquisition of each range data set. If reconstruction is proceeding well, we can assume that the clustering approach is performing as expected and that it may be applied to the acquisition of another view. If, however, reconstruction has seemingly stalled, we can assume that there is a problem and another approach is called for. In that case, we will orient the sensor as far from all previous positions as possible by applying the view history method of Section 5.1. In this way, we can assure that reconstruction of the 3-D model will always continue normally.

A simple way to determine the rate at which reconstruction is proceeding would be to monitor the number of exposed voxel faces in each model after the assimilation of each range image. If the number of voxel faces in the surface model S_i has increased by more than 5% with the most recent view acquisition, or

$$(S_i / S_{i-1}) > 1.05, \tag{6.7}$$

and the number of voxel faces in the occluded model O_i has decreased by more than 5% with the most recent view acquisition, or

$$(O_{i-1} / O_i) > 1.05, \tag{6.8}$$

we can assume that reconstruction is proceeding normally. Thus if either the condition of Eq. 6.7 or the condition of Eq. 6.8 is false, then we will apply the view history approach of Method 1; otherwise, we will apply the clustering approach of Method 4:

$$\begin{aligned} &\text{IF } (((S_i / S_{i-1}) < 1.05) \text{ OR } ((O_{i-1} / O_i) < 1.05)) \\ &\qquad \text{APPROACH} = \text{METHOD 1} \\ &\text{ELSE} \\ &\qquad \text{APPROACH} = \text{METHOD 4.} \end{aligned} \tag{6.9}$$

With these rules, the functions described in Sections 6.1 and 6.2, the criteria for reconstruction termination defined in Section 6.3, and the approaches discussed in Chapter 5, we should now have a completely autonomous and intelligent best-next-view system. A flowchart of the procedure followed by the system is illustrated in Fig. 6.6.

6.5 Software Implementation

For the software implementation of the best-next-view system, we have written a program in the C++ computer language which employs the VolGrid 3-D bitmap class described in the appendix for the manipulation of reconstructed 3-D models. The program has been compiled under both the IRIX 5.3 operating system for Silicon Graphics machines and Sun Solaris 5.5.

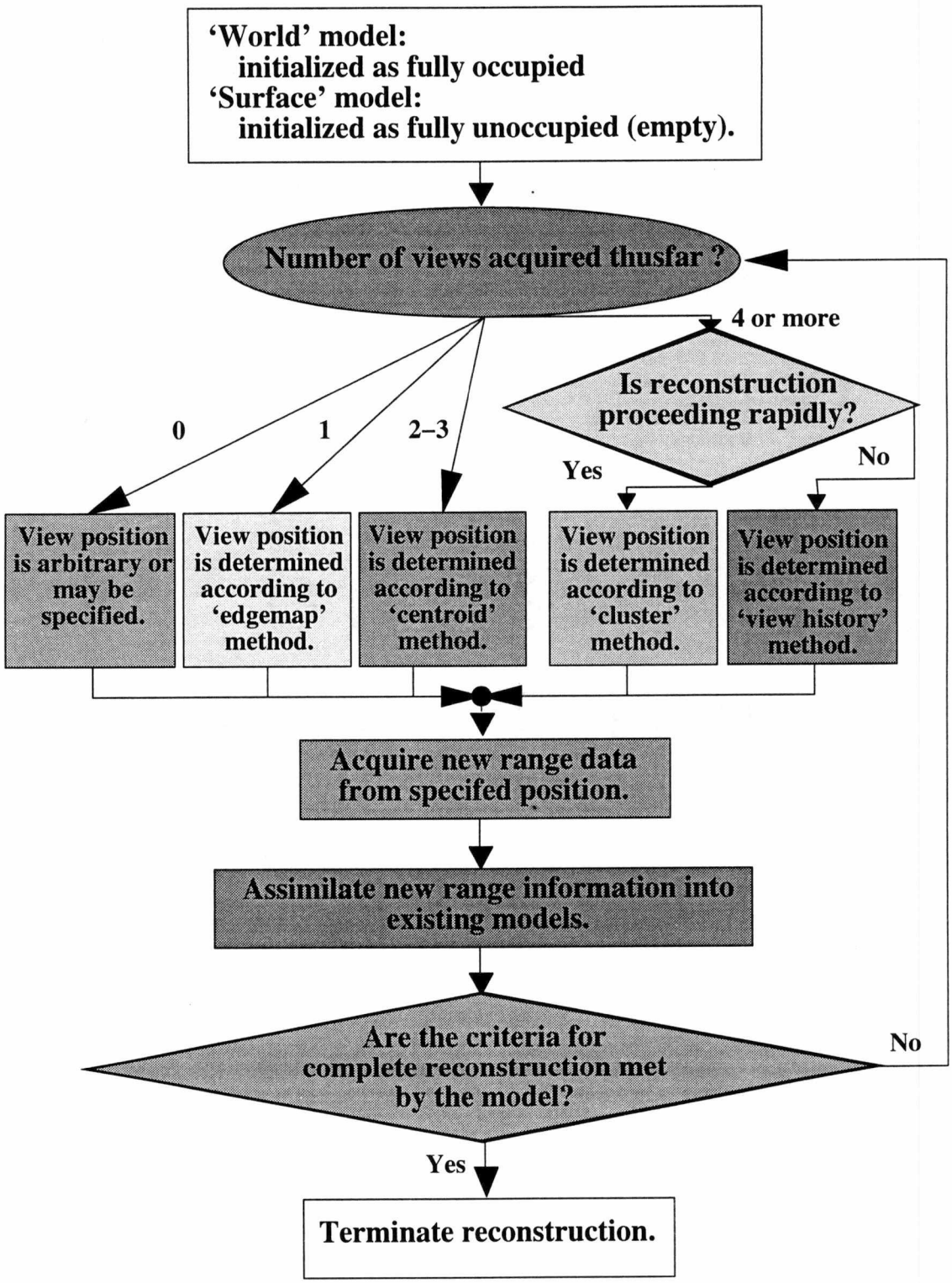


Figure 6.6: *A flowchart of the procedure which the best-next-view system described in this document follows.*

The compiled application can take as command line arguments any of the parameters necessary to continue from the current stage of reconstruction. On the command line, one can specify the name of the file containing the ideal model (Inventor format) or the name of the file containing the reconstructed model. The dimensionalities of the acquired images (always square) and the reconstructed model (always cubic) may be designated, as well as the encoding type of the output image files (color postscript, gif, grey postscript, none, portable pixmap, or IRIS rgb). The application will by default determine the next view position, acquire a range-intensity pair, and assimilate the range information into the prototype model, but is capable of only [a] determining the next view position, [b] acquiring the range-intensity image pair from the specified view position, or [c] reconstructing the model from the most recently acquired range information, dependent on command line arguments.

The user may also specify the parameters controlling the methodology used in reconstructing the prototype model. The **x**, **y**, and **z** coordinates of the next view position may be manually set, or one may specify the method to be used in determining that position (view history, edge map, average or median occluded voxel face location, or centroid of the largest cluster of occluded voxel faces) in determining that position. The user may instruct the system as to how many views should be acquired using the specified method. The option also exists to let the system reconstruct the model as completely as possible. If this option is selected, the model will be reconstructed to a threshold of accuracy at which the model may be considered to be sufficiently reconstructed, as discussed in Section 6.3, and the method used to determine the next sensor pose will be determined by the rules discussed in Section 6.4. One can optionally engage the system to separate closely spaced views, as discussed in Section 6.2. All these options are listed when the name of the application is followed on the unix command line by the option '–h.'

For users more comfortable with a continuous, visible interface with the program, the option also exists for the application to launch a graphical user interface. This interface

was written externally to the 'nextview' C++ code in the Java computer language, most commonly seen in world wide web applications. The Java language was selected for its simplicity, versatility, and multi-media nature. Once the graphical interface is engaged, it can call the 'nextview' application as a native method, but the interface is simply a facade or front end for the actual system. Screen shots of the graphical interface and its four interdependent panels are illustrated in Figs. 6.7 and 6.8.

The graphical interface presents all the same options to the user as the command line for the 'nextview' application. In addition, the user is able to view all range and intensity images acquired thus far from the ideal model (on the Image Panel), review the statistical behavior of the reconstruction by examining the graphs on the Statistics Panel, or spawn an external viewer to visualize either the ideal 3-D model or the reconstructed model (from the Model Panel). Online help is available which contains programmer information, the application's purpose, and documentation regarding the use of the 'nextview' application and its interface.

The Java interface is a fully functional front end for the 'nextview' application. With the 'nextview' application and its interface, we should be able to quickly compile and visualize the results of reconstruction. The complete application spent a reasonably large period of time in development, allowing time for debugging and performance improvement. In the next chapter, we will see the results of the implementation of the system, which will verify the overall effectiveness and worth of the system.

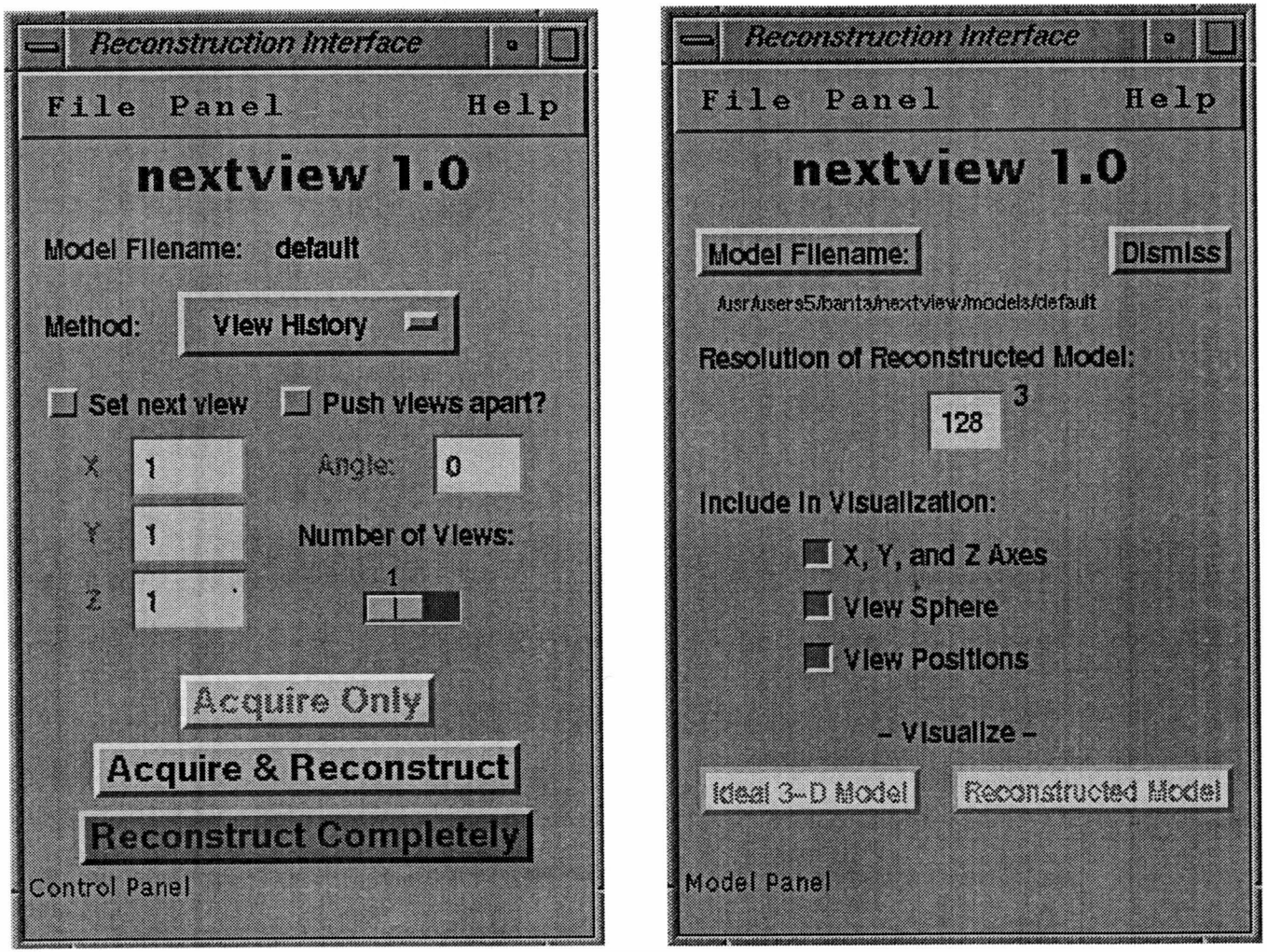


Figure 6.7: *(a) The Control Panel, and (b) the Model Panel of the nextview graphical interface.*

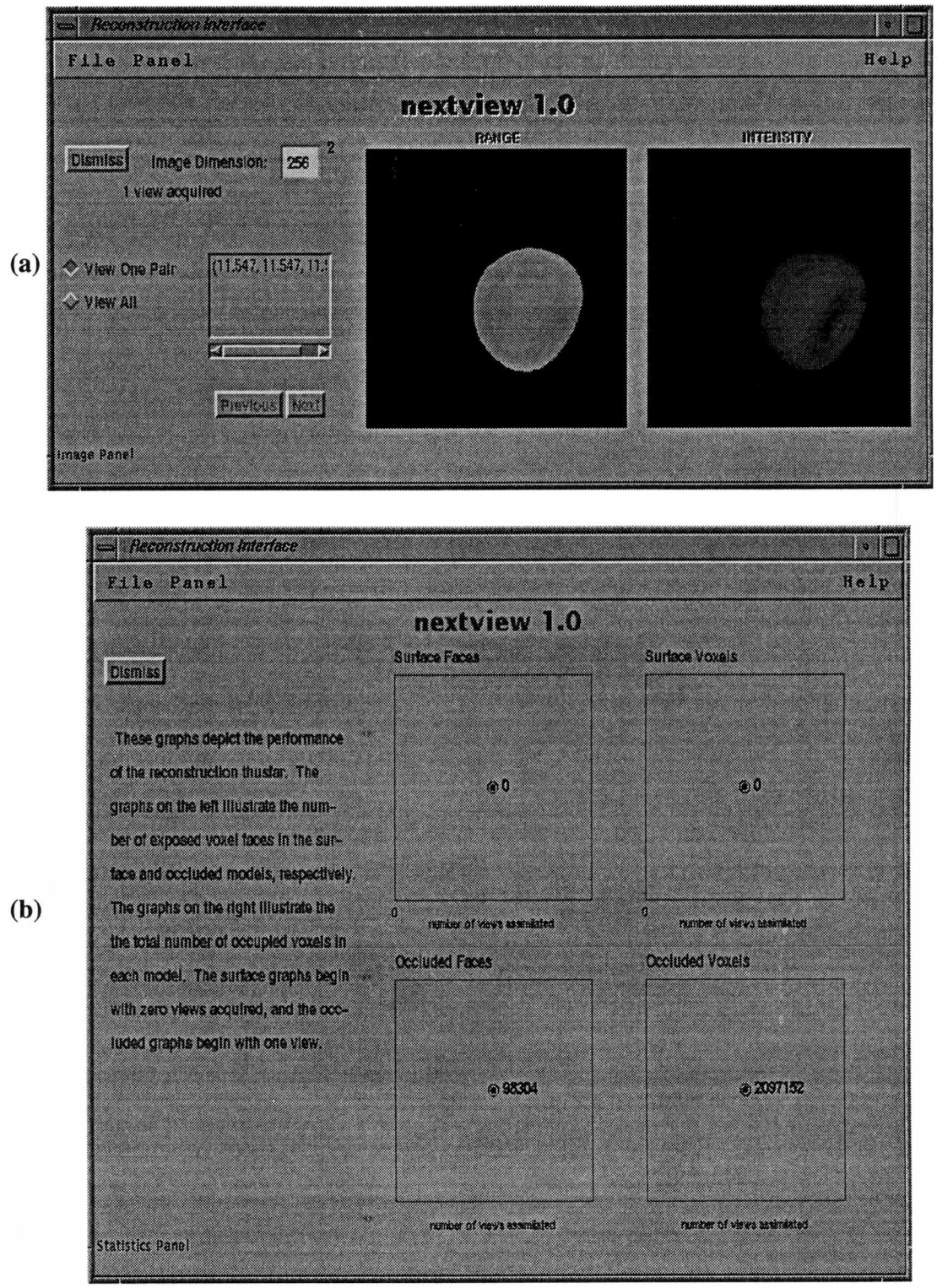


Figure 6.8: *(a) The Image Panel, and (b) the Statistics Panel of the nextview graphical interface.*

CHAPTER 7

EXPERIMENTAL RESULTS

In this chapter, we present results from the propagation of assorted 3-D models through the previously described best-next-view system. As a standard, the images acquired will be of resolution 512×512, and the resolution of the reconstructed models will be 256×256×256. At this level of resolution, the limited complexity of the models we shall present will allow adequate discrete tessellation of the work space, and the time required for calculation and reconstruction should be reasonable (typically between three and five minutes for the determination, acquisition and assimilation of each view on a 150 MHz Silicon Graphics R4400 Indigo2 Extreme workstation, or between two and three minutes on a 195 MHz Silicon Graphics R10000 Indigo2 Impact/TRAM workstation).

We will present results from the reconstruction of six models: the pipe model, the mug model, the vise model, the valve model, the chair, and the hand model. For the first input model (the pipe model), we will individually examine the results from each view acquisition, illustrating the best-next-view system at work and the effects of the assimilation of each range data set. For the remainder of the models, we will present a concise set of results, illustrating the reconstructed model only after all range data sets have been assimilated. The first four models were reconstructed at a resolution of 256×256×256, while the chair model was reconstructed at a higher resolution of 400×400×400 to illustrate the diversity of the system. The final model, the hand model, was reconstructed beyond the threshold of accuracy determined in the previous chapter so that we might examine an *absolutely* reconstructed model.

7.1 The Pipe Model

The pipe model, illustrated in Fig 7.1, is comprised of an array of cylinders arranged in a grid pattern. Because of their rounded shape, cylinders might present a challenge for our discrete best-next-view system. Also, the cylinders are arranged such that the model will contain many self-occluding contours, which might lead a lesser system to orient the sensor on previously sampled regions of the object surface. Thus this model should serve to verify the validity of the function we have written for providing an unoccluded view of a target vicinity on an object.

The first range and intensity images of the pipe model, shown in Fig. 7.2, were acquired from an arbitrary position on the **x** coordinate axis. A voxel rendering of the recovered pipe model after the assimilation of the first range data set is shown in Fig. 7.3. In this illustration, the dark shaded surface represents recovered surface data, and the brighter shaded surface represents the information that was occluded from the sensor's view. We will continue reconstruction from this model, the method used to determine the next view position decided according to the discussion of section 5.4.

The second view position will be determined by the edge-based sensor placement

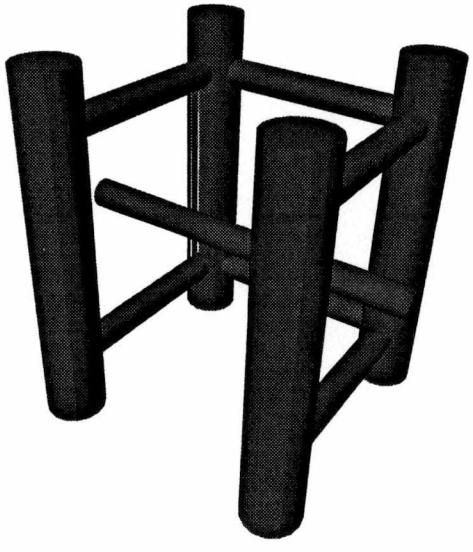


Figure 7.1: *The ideal pipe model.*

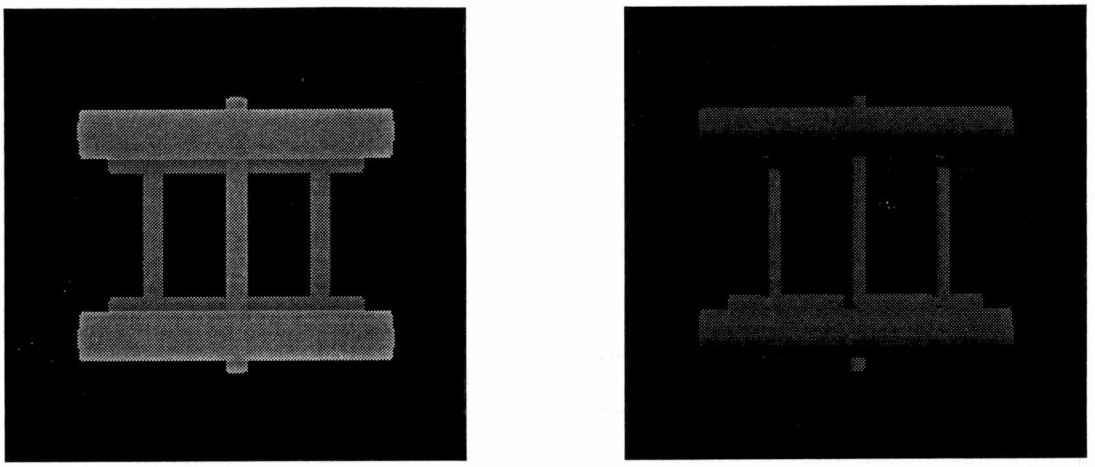


Figure 7.2: *The range and intensity images acquired from the first view position for the pipe model.*

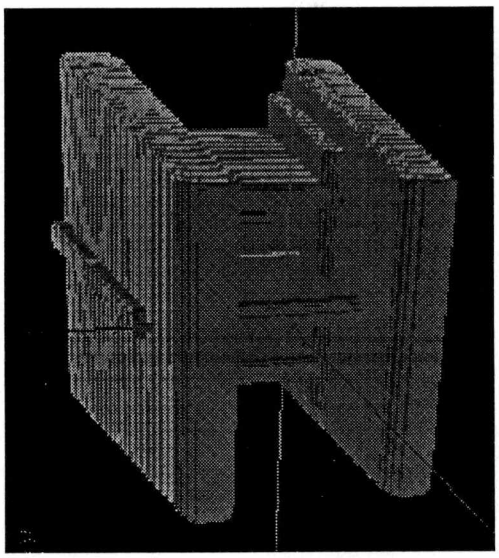


Figure 7.3: *A voxel rendering of the recovered pipe model after the assimilation of the first range data set.*

technique of section 5.2.2. The image resulting from the differentiation of the range image of Fig. 7.2 is shown in Fig. 7.4. In this figure is highlighted the center of the image and the brightest pixel average, which together define a vector that, when transformed to three dimensions, will be used to define the best-next-view position. The range and intensity images acquired from the second view position are shown in Fig. 7.5, and a voxel rendering of the recovered model after the assimilation of the second range data set is shown in Fig. 7.6. We notice that the amount of occluded information has been greatly reduced between the first and second views. This would probably be true of any two view positions separated by a reasonable angle, but because these two positions are necessarily orthogonal, the amount of new information recovered is truly significant.

The third view position was then determined using the occluded surface median approach described in section 5.3.2. Examining the surface of the occluded model, one might predict where the centroid of the surface might be and the approximate orientation of the overall surface normal. However, one cannot predict whether a vector radiated from the surface centroid in the direction of the surface normal will be occluded; thus we cannot truly predict the location of the best-next-view position. In Fig. 7.7 are illustrated the range and intensity images acquired from the third view position for the pipe model. Note that

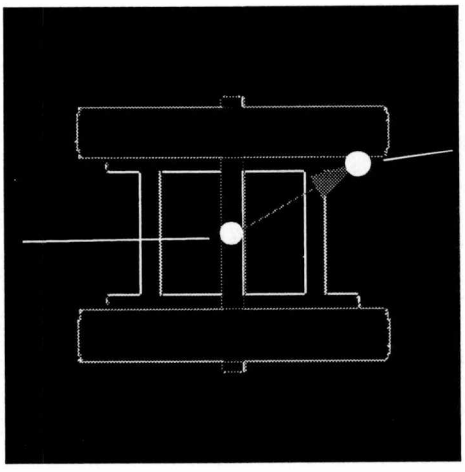


Figure 7.4: *The image resulting from the differentiation of the range image in Fig. 7.2.*

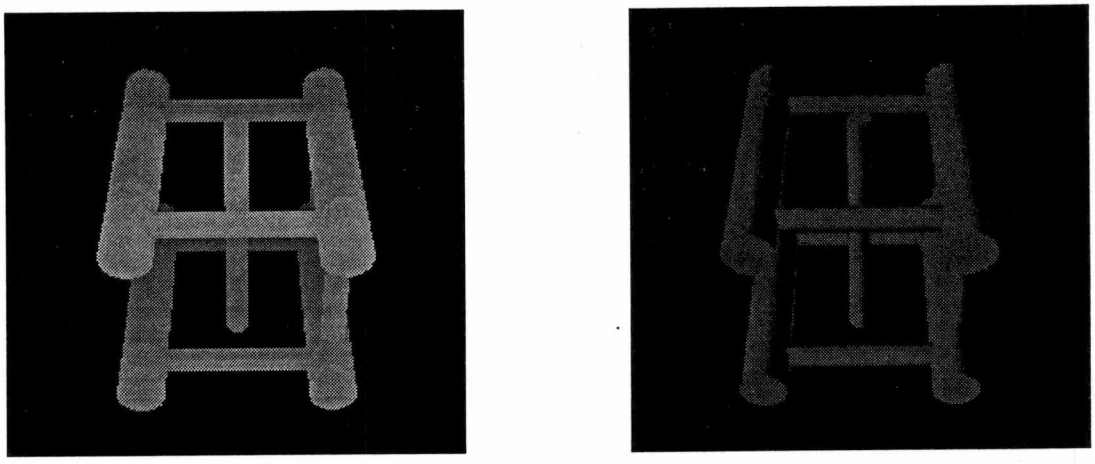


Figure 7.5: *The range and intensity images acquired from the second view position for the pipe model.*

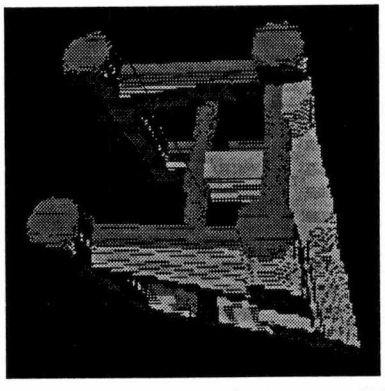


Figure 7.6: *A voxel rendering of the recovered pipe model after the assimilation of the second range data set..*

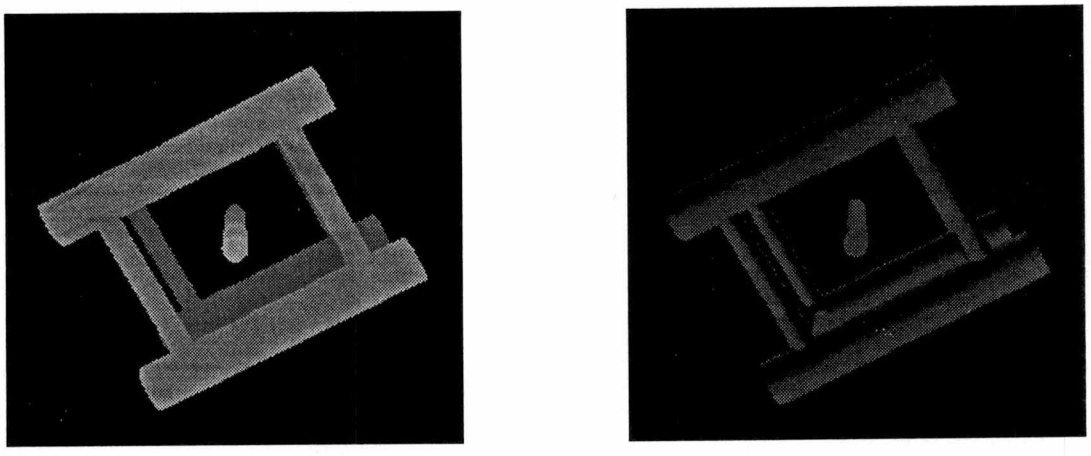


Figure 7.7: *The range and intensity images acquired from the third view position for the pipe model.*

the third view position is nearly orthogonal to both the first and second view positions. For this reason, the amount of new information recovered from the third range data set was also fairly significant.

As we can see from the voxel rendering of the recovered model in Fig. 7.8, reconstruction for this model is nearing completion; the remainder of the occluded data should be selectively surveyed to determine the next view position. For this model, the occluded data has been broken into several disjoint surface patches at an early stage, which would lead us to select the clustering method to determine the next view position. However, because the best-next-view system is automating the reconstruction process, the method to be used will again be the occluded surface median approach. In this case, the model is being reconstructed more rapidly than we originally expected; however, we would not expect such behavior for a more complex model.

The occluded surface centroid approach actually successfully oriented the fourth sensor position on the two large clusters of occluded data visible in Fig. 7.8. In Fig. 7.9 are shown the range and intensity images acquired from the specified position, and a voxel rendering of the recovered pipe model after the assimilation of the fourth range data set is illustrated in Fig. 7.10. The model after four view acquisitions indeed does appear to be

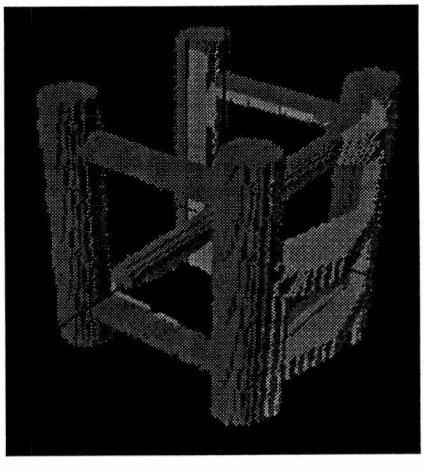


Figure 7.8: *A voxel rendering of the recovered pipe model after the assimilation of the third range data set.*

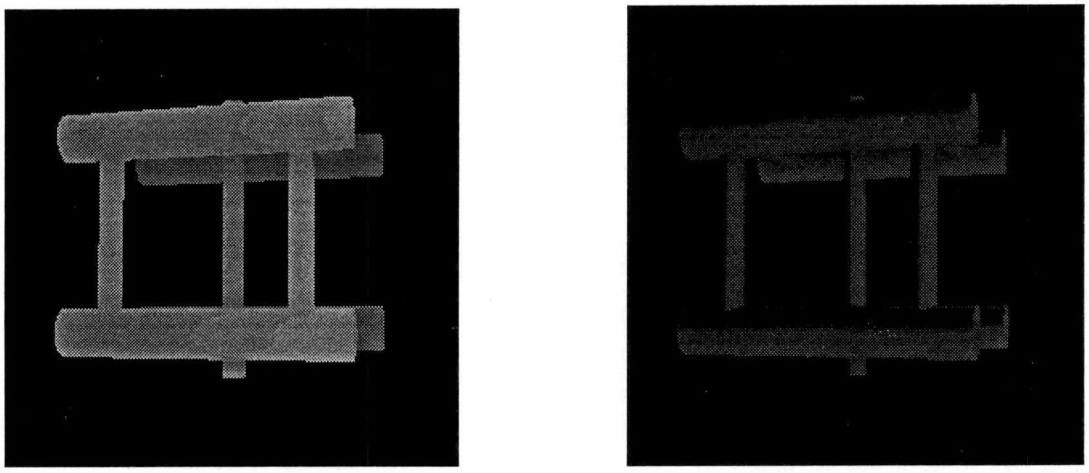


Figure 7.9: *The range and intensity images acquired from the fourth view position for the pipe model.*

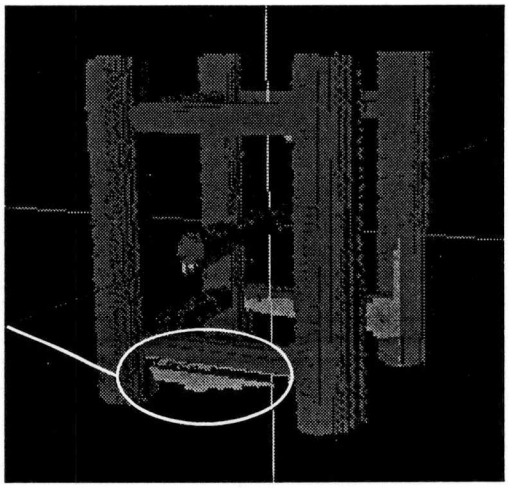


Figure 7.10: *A voxel rendering of the recovered pipe model after the assimilation of the fourth range data set, with the largest occluded cluster circled in white..*

fairly complete. Only a small number of scattered occlusions remain, and should probably be recovered with the next view acquisition.

Because reconstruction has been proceeding rather rapidly, the best-next-view system will determined the location of the fifth view position using the occluded surface clustering approach (as discussed in Chapter 5). The results of clustering the occluded voxel surface faces are shown in a voxel rendering in Fig. 7.11, and are circled in white in Fig. 7.10. The next view position was selected by the system to eliminate this occluded cluster, and the images acquired from that position are shown in Fig. 7.12. In Fig. 7.13, a voxel rendering of the reconstructed model after the assimilation of this range data set is shown from the same perspective as that of Fig. 7.10. Obviously, the cluster on which the fifth view was oriented was eliminated by the assimilation of this range data set, and it appears from Fig. 7.13 that any occlusions that remain in the model are sparse. After the assimilation of the fifth range data set, the model was determined by the best-next-view system to be adequately reconstructed.

Now that the best-next-view system has automatically reconstructed a prototype of the pipe model, we may examine the collective results. In Fig. 7.14, we see the ideal pipe model illustrated with each of the positions from which the system acquired range data. From this figure, one may get a perspective of the images from each view position.

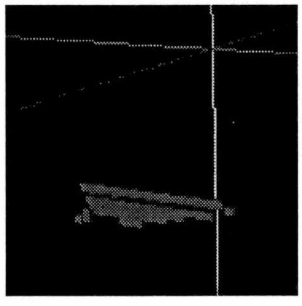


Figure 7.11: *A voxel rendering of the results of clustering the voxel surface faces in the occluded model of Fig. 7.8.*

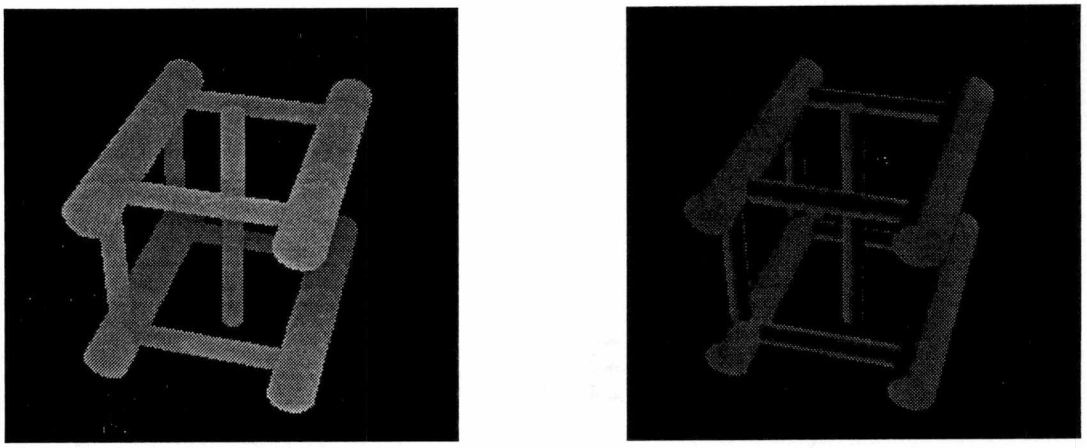


Figure 7.12: *The range and intensity images acquired from the fifth view position for the pipe model.*

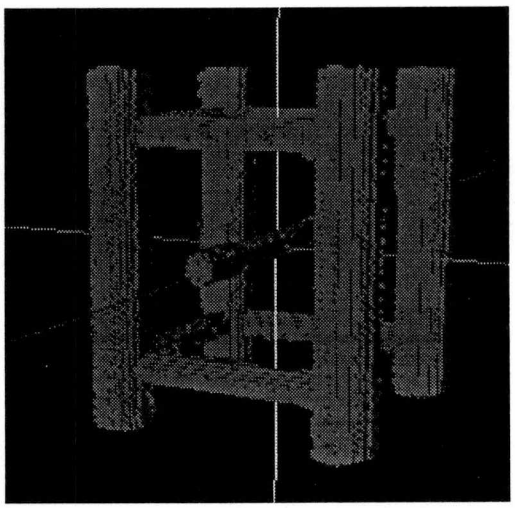


Figure 7.13: *A voxel rendering of the recovered pipe model after the assimilation of the fifth range data set.*

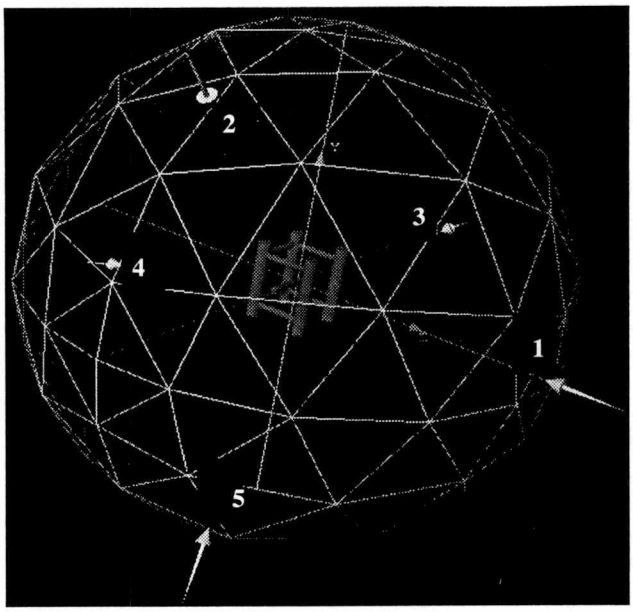


Figure 7.14: *The ideal pipe model shown with the positions from which the best-next-view system acquired range and intensity images (represented as arrows).*

In Fig. 7.15 are illustrated four graphs which depict the iterative behavior of the best-next-view system. The graphs on the left illustrate the number of exposed voxel faces in the surface and occluded models, while the graphs on the right illustrate the total number of voxels in each model. The surface model graphs (top) show that the surface model does increase in size, as expected, with the assimilation of each range data set, until the final images were acquired and reconstruction was terminated. The occluded model graphs (bottom) illustrate the dwindling size of the occluded model with each view. While the number of exposed occluded voxel faces converges toward zero, the number of occluded voxels will always converge to some finite positive number. These are the voxels *inside* of the model.

In Fig. 7.16, we see a surface rendering of the model reconstructed after the assimilation of the five range data sets compared with the ideal pipe model. One can clearly see that the reconstructed model strongly resembles the ideal model, proving the validity of the reconstruction system. The surface of the reconstructed model is of course not as smooth as the surface of the ideal model; because we are representing our volume

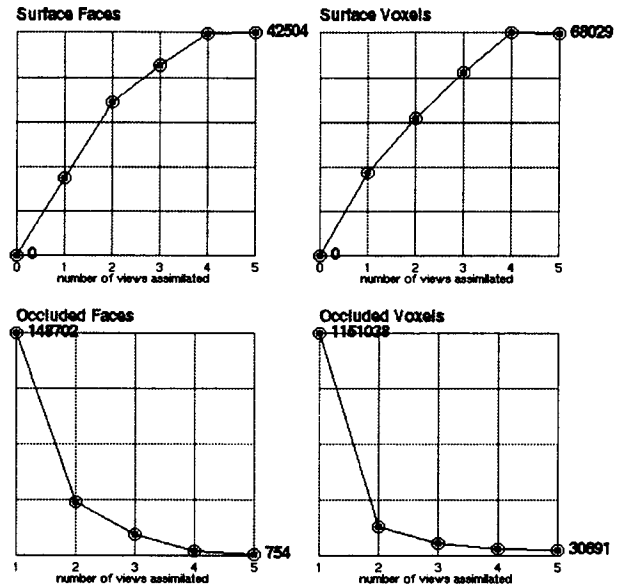


Figure 7.15: *Four graphs illustrating the change in number of surface voxel faces, surface voxels, occluded voxel faces, and occluded voxels for the reconstructed pipe model with the assimilation of each range data set.*

data as binary (occupied or empty) rather than as probabilistic grayscales, a continuous surface might become somewhat mangled during the reconstruction process. Regardless of the texture of the surface, the reconstructed model indeed does visually compare well with the ideal model, allowing us to move on to other models.

7.2 The Mug Model

In this section we will focus our attention on a model of a coffee mug. The mug model, illustrated in Fig. 7.17, should be an interesting model for us to reconstruct, for two reasons. The first is obvious: the model is that of an actual physical object, with which one is probably familiar and can freely associate. The second reason stems from the nature of the best-next-view system. Examining the mug model, one will notice that a large percentage of the surface of the model lies inside of a concavity. The significance of this fact with respect to our best-next-view system is that the edge of a concavity will tend to

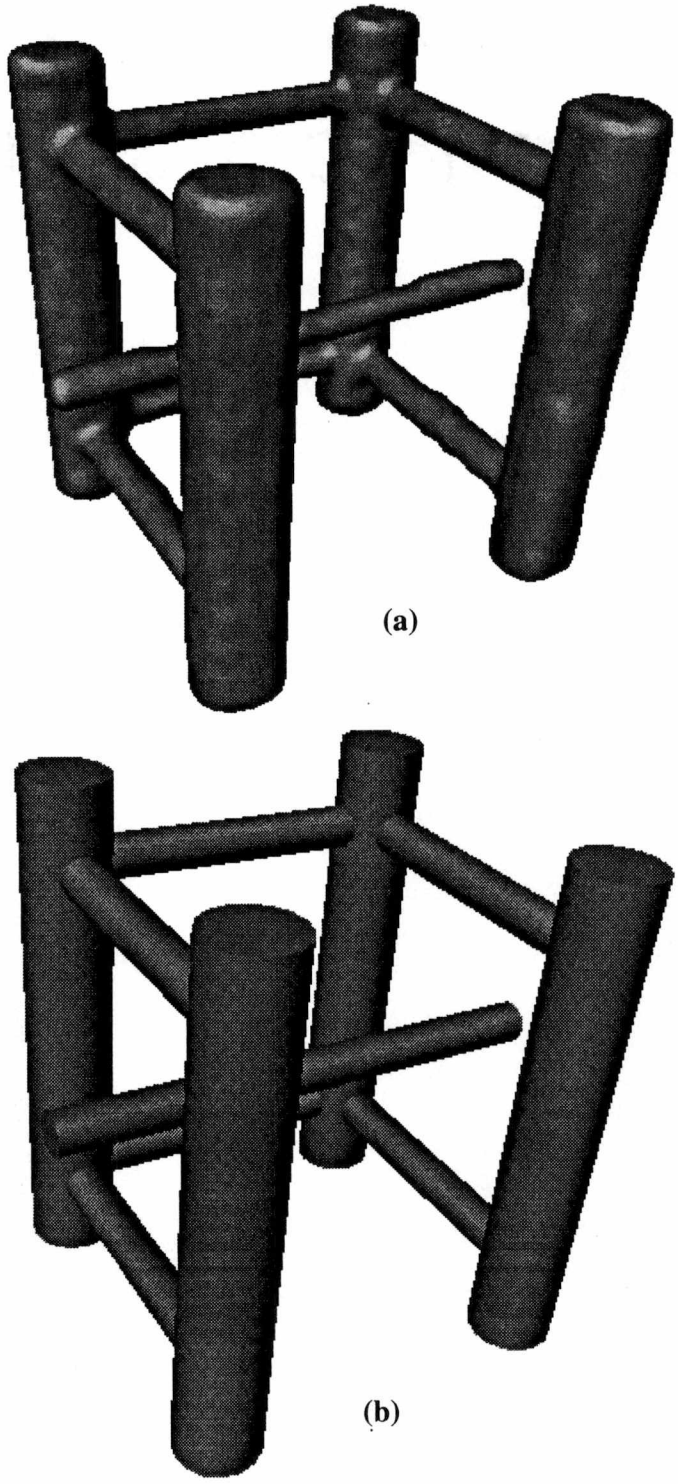


Figure 7.16: *(a) A surface rendering of the reconstructed pipe model, compared with (b) the ideal pipe model.*



Figure 7.17: *The ideal mug model.*

to occlude the interior of the concavity when viewed from most angles.

The test of the best-next-view system will be in determining the position and orientation from which the interior of the concavity may be observed. While the holistic view-determining techniques use the normal of the surface of interest to determine the view orientation (and thus might tend to position the sensor where only the edge of a concavity is visible), our system should be able to provide the sensor with an unoccluded view of an object's surface (as discussed in section 5.2.1). Both the interiors of the mug and its handle should serve to verify this function.

We will directly examine the final results of the submission of the mug model to the system for complete reconstruction. In Fig. 7.18 we see a compilation of the seven range and intensity images acquired during the reconstruction of the mug model, and in Fig. 7.19, the ideal mug model is illustrated with the view positions shown relative to the model. It appears that the majority of the views were acquired in an attempt to view the occluded information inside the mug, as we predicted. The first view assimilated the preliminary profile information. The second view appeared to be examining the mug handle and the top of the cup. With the third view acquisition, the bottom surface of the cup was assimilated. The remainder of the views were apparently acquired with the intent of viewing the interior of the mug and its inner surface. An isosurface rendering of the reconstructed mug model is compared with the ideal mug model in Fig. 7.20.

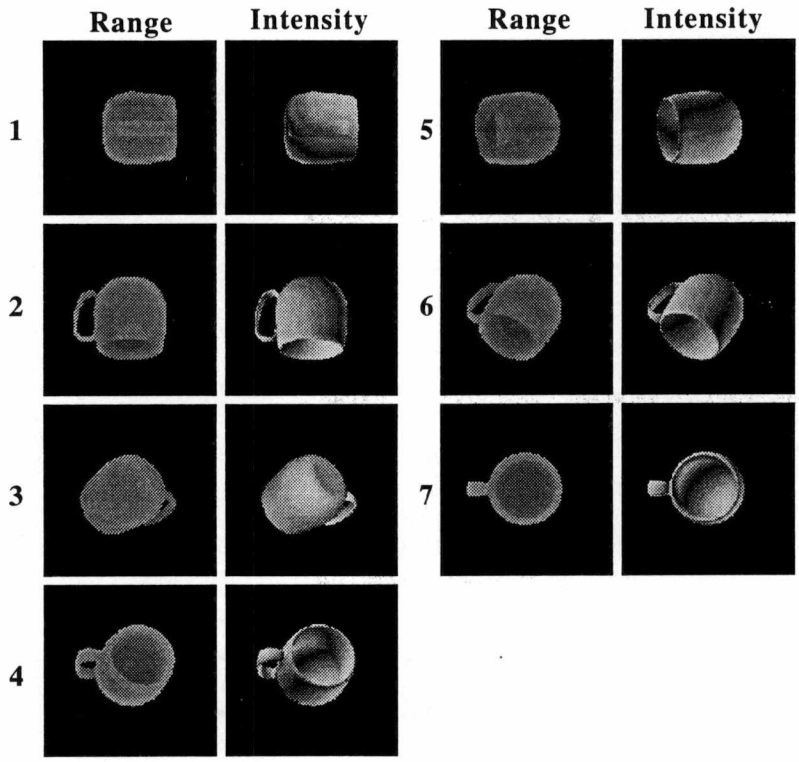


Figure 7.18: *The seven range and intensity images acquired for the mug model.*

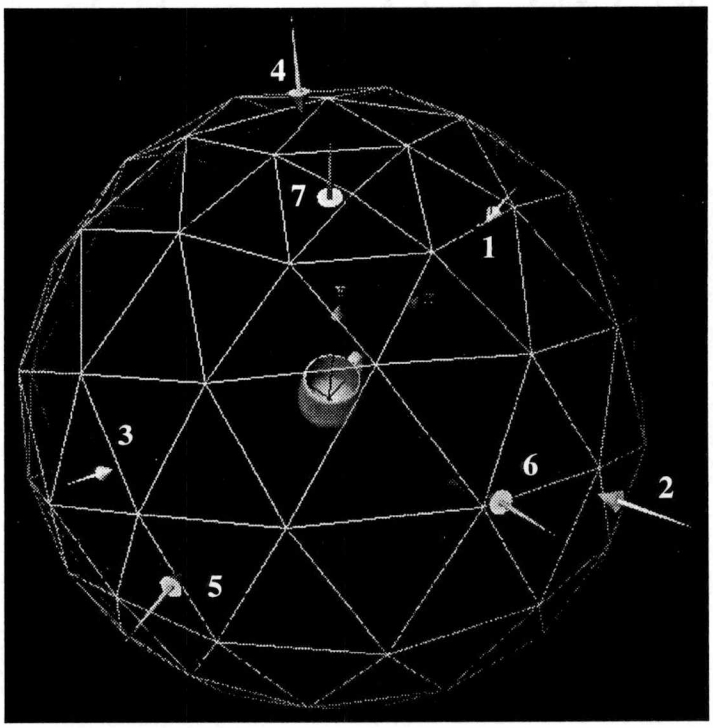


Figure 7.19: *The ideal mug model illustrated with the seven view positions.*



(a)



(b)

Figure 7.20: *(a) A surface rendering of the reconstructed mug model, compared with (b) the ideal mug model.*

Seven view acquisitions may certainly be considered reasonable for the reconstruction of a somewhat complex model. Considering the different orientations of all the surfaces in the mug model, it is surprising that more views were not required. The entirety of the outside surface and the interiors of the mug and its handle all must be assimilated into the model for adequate reconstruction. We can see that the reconstructed model of Fig. 7.20 (a) indeed does compare well with the ideal model in Fig. 7.20 (b). The concavity of the interior of the mug appears to be intact, and the shapes of the handle and the cup itself were reconstructed correctly. With seven views, the mug model has been reconstructed to an adequate level.

7.3 The Vise Model

The next model we shall attempt to reconstruct will be that of an industrial vise. The ideal 3-D model of the vise is shown in Fig. 7.21. The model is a reasonably accurate reproduction of a vise, consisting of a handle, the clamp of the vise, and the knob which tightens or loosens the vise. The vise model should introduce an industrial motive for the application of a best-next-view system. The model is simultaneously simple-looking and somewhat complex. The most difficult part of an object for a best-next-view system to

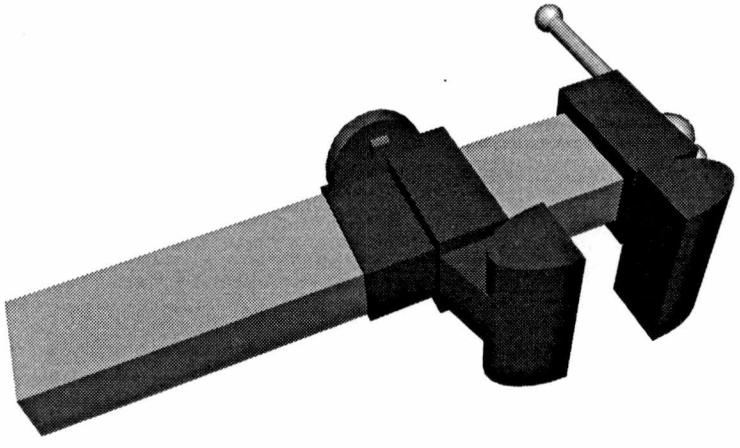


Figure 7.21: *The ideal vise model.*

reconstruct is the intricate details, such as the bolt on the clasp of the vise or the rounded tip of the tightening knob. This is the point at which the resolution of the model becomes important: if some aspect of an object is smaller than the sampling resolution, it will obviously not be assimilated into the model.

Only four images were required to adequately reconstruct a prototype of the vise model. The range and intensity images acquired are shown in Fig. 7.22. The view positions relative to the model are shown in Fig. 7.23. The model reconstructed from the four range views is illustrated in Fig. 7.24 (a), which may be compared with the ideal vise model, shown in Fig. 7.24 (b). We will note that the total number of views required for reconstruction was minimal. It appears from the reconstructed model, however, that certain details of the model were not assimilated into the prototype, specifically the bolts on the side of the vise and the indented tips of the tightening knob. Fortunately, this observation is false. The information of the bolts and the indented tips is contained in the model, but it was smoothed out for the purpose of visualization. Thus the surface transitions in the reconstructed model are simply more gradual, and hence less visible. The vise model was thus successfully reconstructed by the best-next-view system.

7.4 The Valve Model

The valve model, shown in Fig. 7.25, is a replication of a cylindrical container with a pipe and valve extruding from it. The model is intended as a reproduction of a component of an actual facility contaminated with radioactivity which is scheduled to be disassembled. Because the radioactivity makes it hazardous for humans to enter the facility for extended periods of time, the dismemberment of the facility will optimally be automated. Thus this model helps to illustrate an application of this system to the handling of hazardous material.

The reconstruction of this model also required only four view acquisitions. The range and intensity images acquired during reconstruction are illustrated in Fig. 7.26, and the

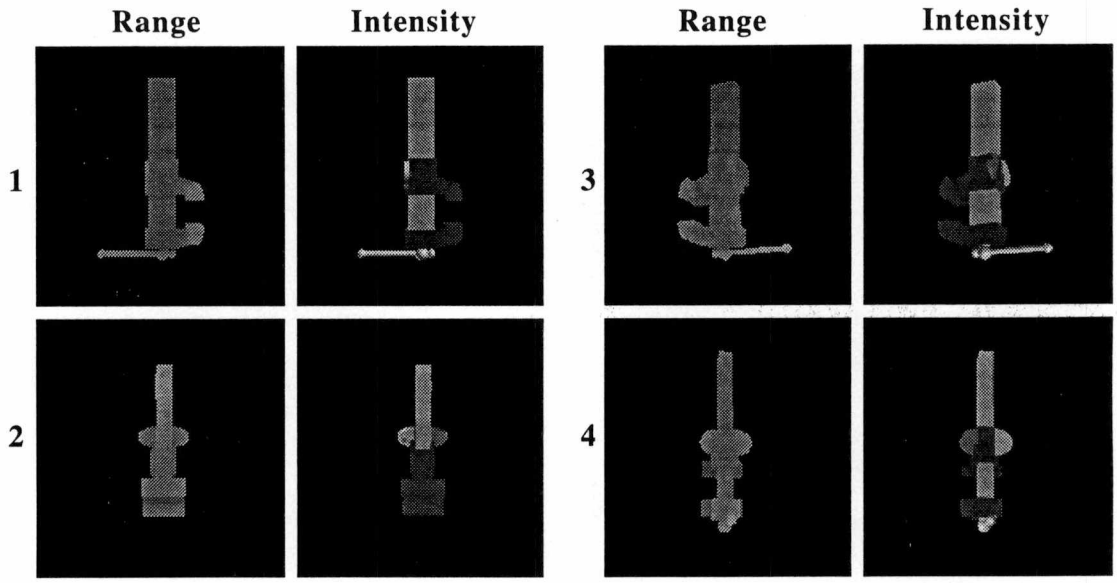


Figure 7.22: *The four range and intensity images acquired for the vise model.*

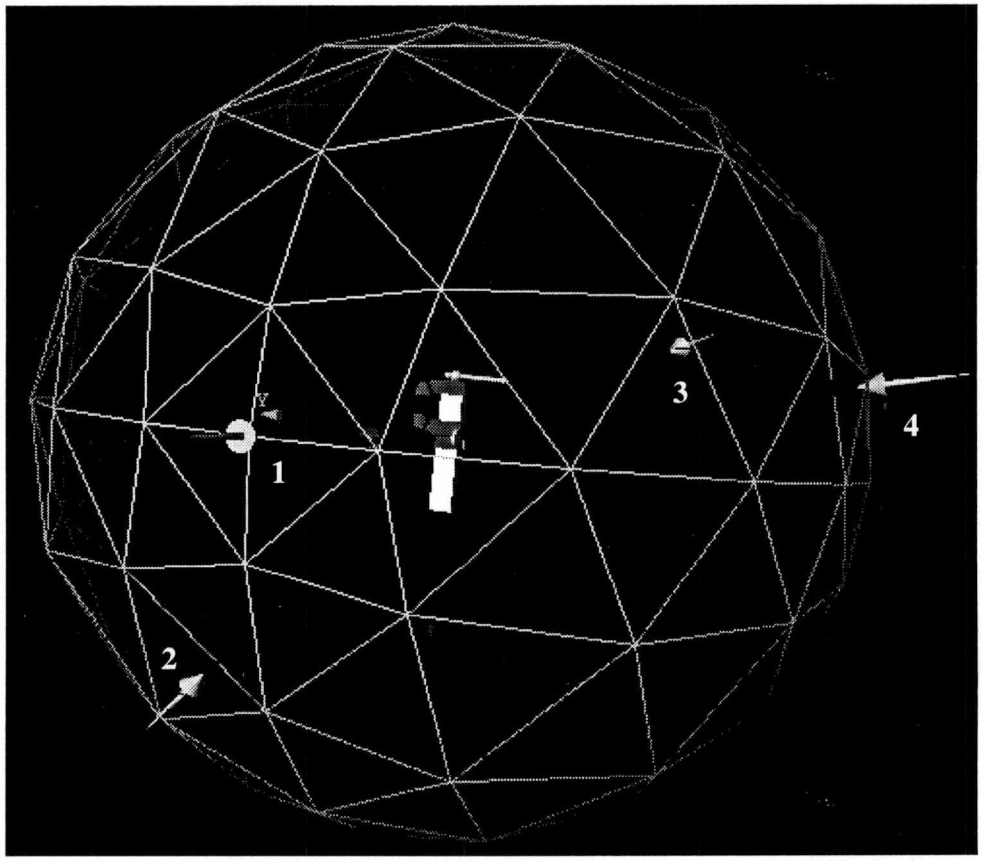


Figure 7.23: *The ideal vise model illustrated with the four view positions.*

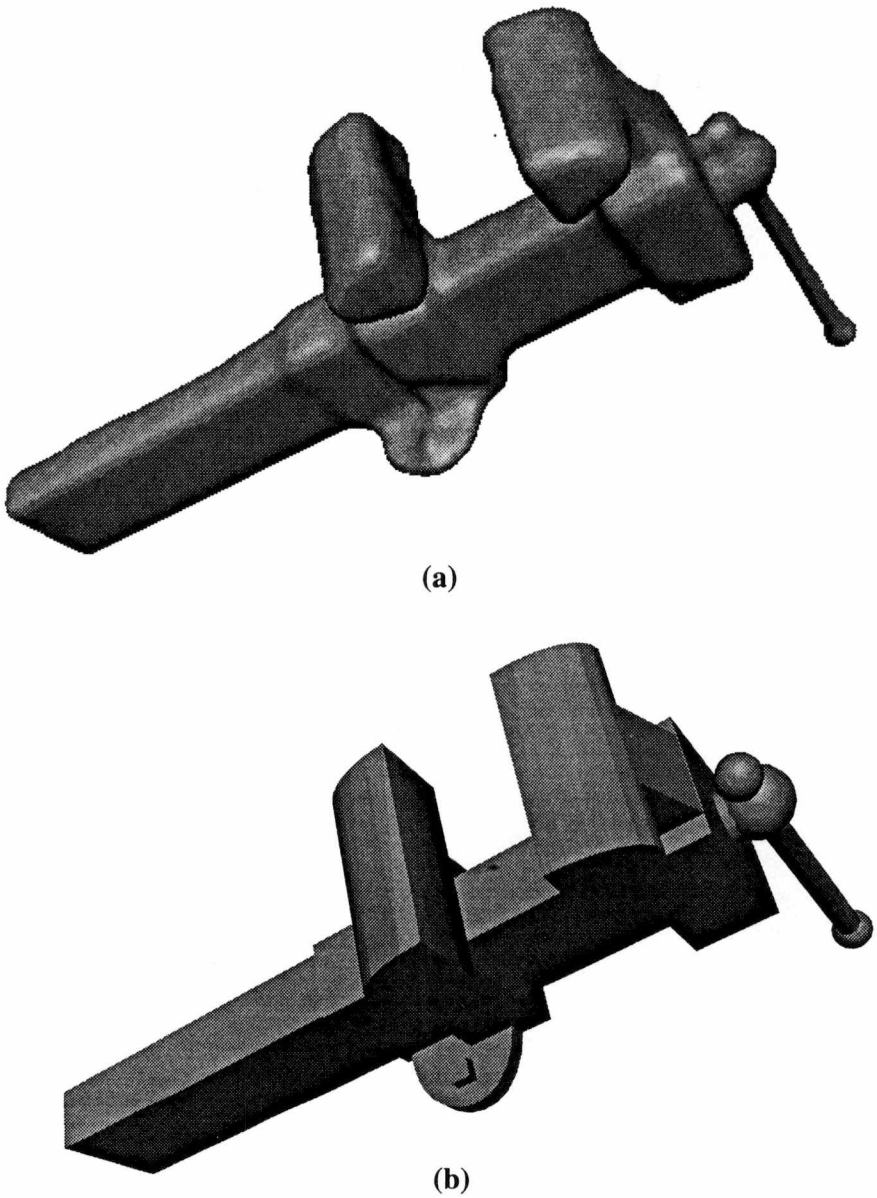


Figure 7.24: *(a) A surface rendering of the reconstructed vise model, compared with (b) the ideal vise model.*

view positions relative to the 3-D model are shown in Fig. 7.27. One will notice that the positions appear to be very well distributed about the model. In the case of this model, this distribution is the most appropriate for adequate reconstruction. However, we will recall the reconstruction of the mug model, and the distribution of views shown in Fig. 7.19. For the mug model, a number of views were required for the interior surface of the mug. Here we find an example illustrating why the same distribution of views will not reconstruct every model equally well.

An isosurface rendering of the reconstructed valve model is compared with the ideal valve model in Fig. 7.28. The effects of smoothing the prototype for visualization are plainly visible: the two parallel plates near the top of the model appear to be almost blended together, and anywhere a discrete boundary between two surfaces is found in the ideal model, a rounded, continuous boundary will be found in the surface rendering of the reconstructed model. Otherwise, the reconstructed model compares well with the ideal, offering a minimum of contrast to the visual survey.

7.5 The Chair Model

The ideal chair model, shown in Fig. 7.29, was also used to illustrate the utility of the image acquisition function in Fig. 3.3. The specific reason the chair model was selected for reconstruction was to test the system's ability to reconstruct objects which are relatively thin in comparison to the overall object dimensionality. The chair model presents us with several instances of such an object in the "boards" which comprise the chair. The chair model might present a challenge to the system due to the void spaces between the boards, which will offset the overall surface normal somewhat. So that the best model will be reconstructed, and to illustrate the versatility of the system, the model will be reconstructed at an increased resolution of 400×400×400.

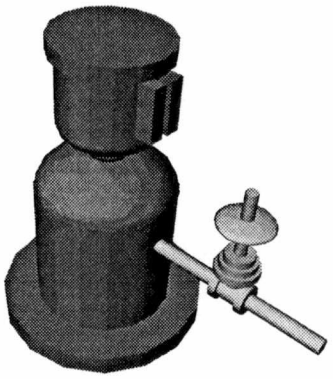


Figure 7.25: *The ideal valve model.*

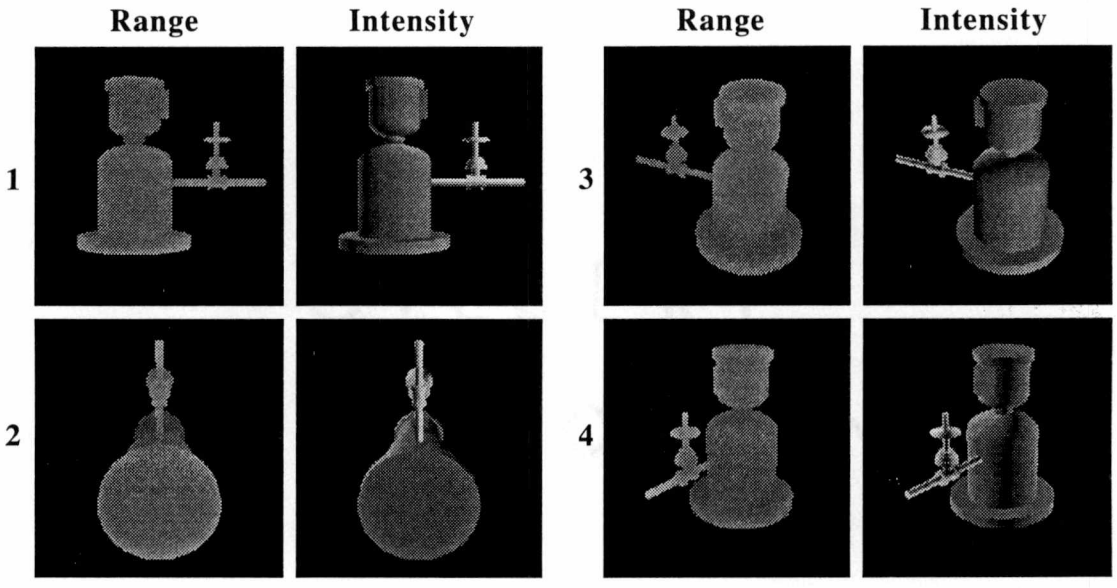


Figure 7.26: *The four range and intensity images acquired for the valve model.*

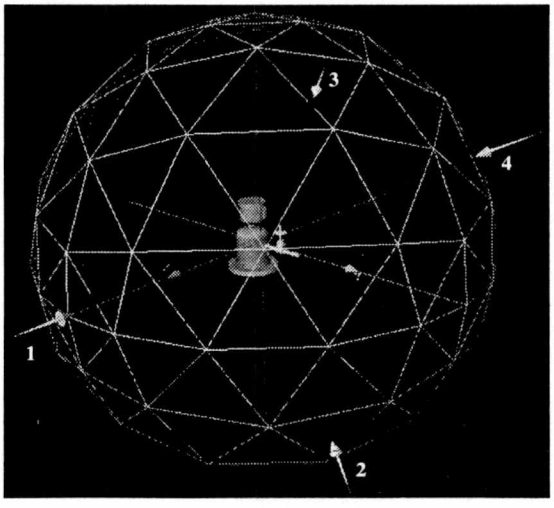
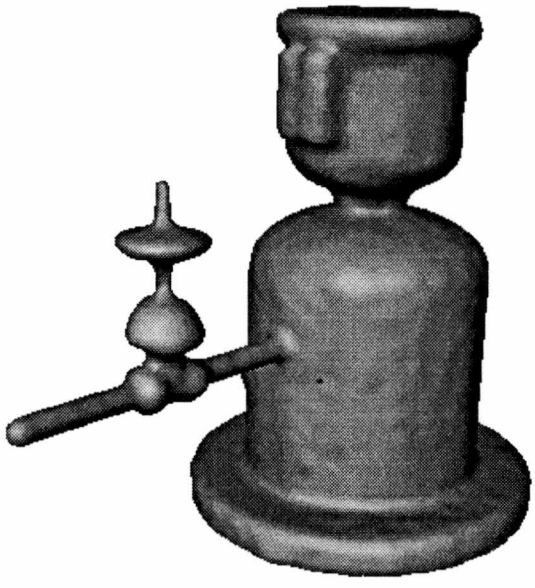
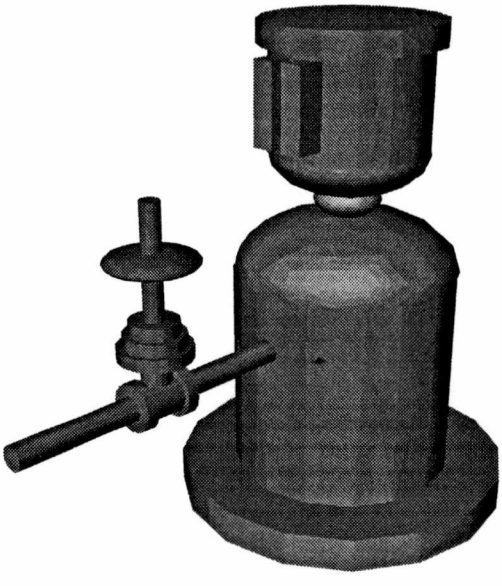


Figure 7.27: *The ideal valve model illustrated with the four view positions.*



(a)



(b)

Figure 7.28: *(a) A surface rendering of the reconstructed valve model, compared with (b) the ideal valve model.*



Figure 7.29: *The ideal chair model.*

Five views were required to sufficiently reconstruct a prototype of the chair model. The range and intensity images from each view are shown in Fig. 7.30, and the positions from which the images were acquired are shown in Fig. 7.31. It appears from Fig. 7.31 that after the assimilation of the first image, each of the remainder of the views was closely oriented to the lateral direction of one or more of the boards comprising the chair model. This is no doubt due to the side surface of each board, which would of course contribute to the overall surface normal.

In Fig. 7.32 we see a voxel rendering of the reconstructed prototype from the chair model compared with the ideal chair model. Due to the increased resolution of the model, it was not possible to generate an isosurface rendering of the reconstructed model (the isosurface function can only have as input a grid of dimension up to 256×256×256). Once more, the reconstructed model compares well with the ideal. However, there appears to be some disparity between the models near the base of the rear boards of the chair. The surface of the boards in the reconstructed model has been eroded away somehow. The origin of this fault may be found in the thickness of the boards. This erosion effect will be discussed further in the next chapter.

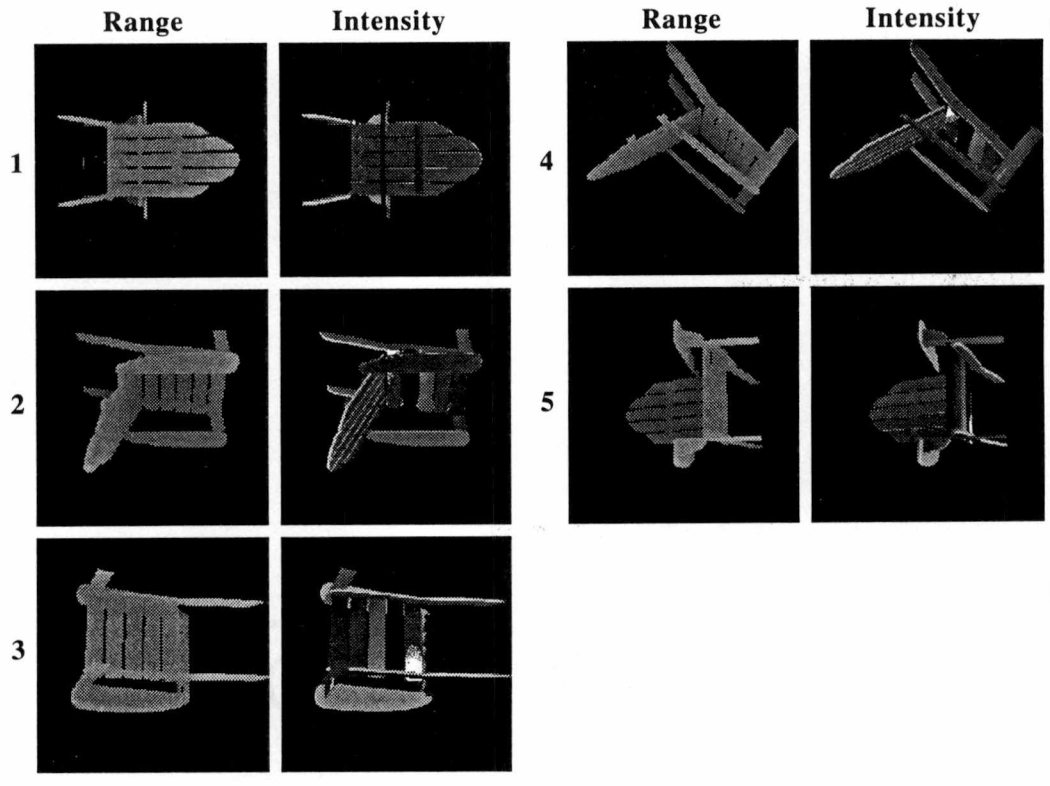


Figure 7.30: *The five range and intensity images acquired for the chair model.*

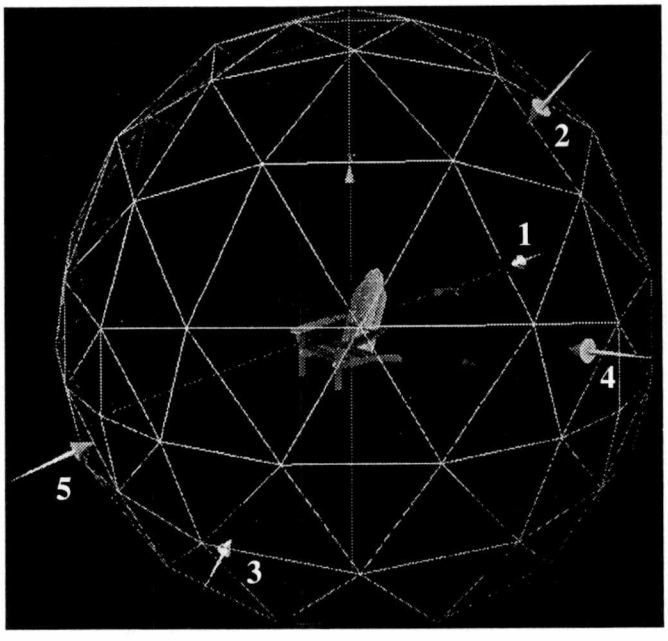


Figure 7.31: *The ideal chair model illustrated with the five view positions.*



Figure 7.32: *(a) A voxel rendering of the reconstructed chair model, compared with (b) the ideal chair model.*

7.6 The Hand Model

The ideal hand model is illustrated in Fig. 7.33. This model is a replication of a robotic manipulator with three finger-like appendages and a semi-spherical base. Located in the center of the robot arm are three cylindrical concavities, which will no doubt provide a challenge to our reconstruction system. The near-negligible thickness of each finger appendage with respect to the overall size of the model might introduce an additional obstacle for the system to overcome.

After the acquisition and assimilation of four range views, the criteria for sufficient accuracy were met by the reconstructed model. The relative positions from which the four range images were acquired are shown in Fig. 7.34, and a voxel rendering of the reconstructed hand model is illustrated in Fig. 7.35. While the view positions appear to be well-distributed about the object and the criteria for sufficient reconstruction were met, it appears that the cylindrical concavities in the center of the arm were not correctly reconstructed, and a fair amount of occluded information remains to be assimilated near the base of the fingers. For this reason and so that we could illustrate the extensibility of the system, four more range images were acquired from positions determined using the view history approach, with the range information iteratively assimilated into the reconstructed model.

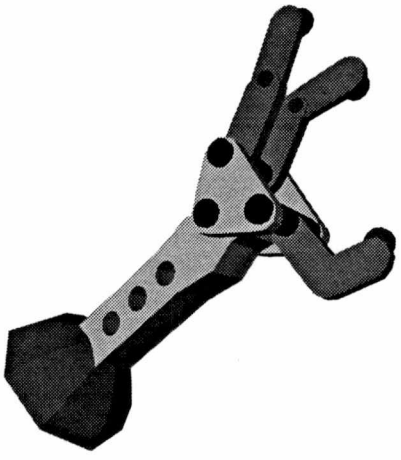


Figure 7.33: *The ideal hand model.*

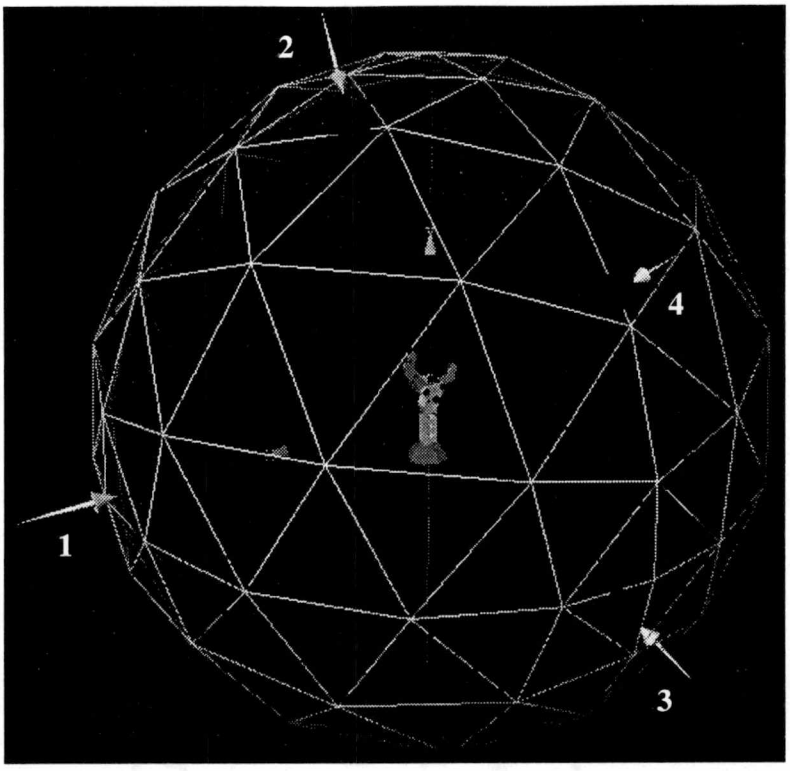


Figure 7.34: *The first four positions from which range data was acquired from the hand model.*

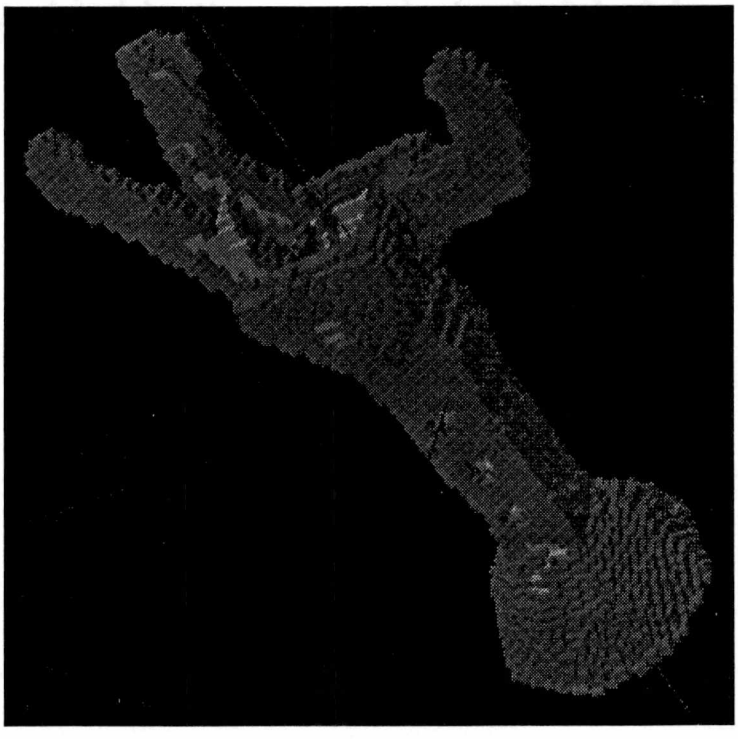


Figure 7.35: *A voxel rendering of the hand model after four range image assimilations.*

The complete set of eight range images which were acquired for the reconstruction of the hand model are shown in Fig. 7.36. The position from which each range and intensity image was acquired is illustrated relative to the ideal hand model in Fig. 7.37. In Fig. 7.38 are shown the four graphs also depicted for the pipe model in Fig. 7.15, which follow the behavior of the reconstructed model as reconstruction progresses. Plainly the size of the occluded model decreases to the internal voxel count of the model, while the size of the surface model appears to peak at a maximum value and gradually trail off. This is explained by those voxels in the surface model which are essentially "false positives":

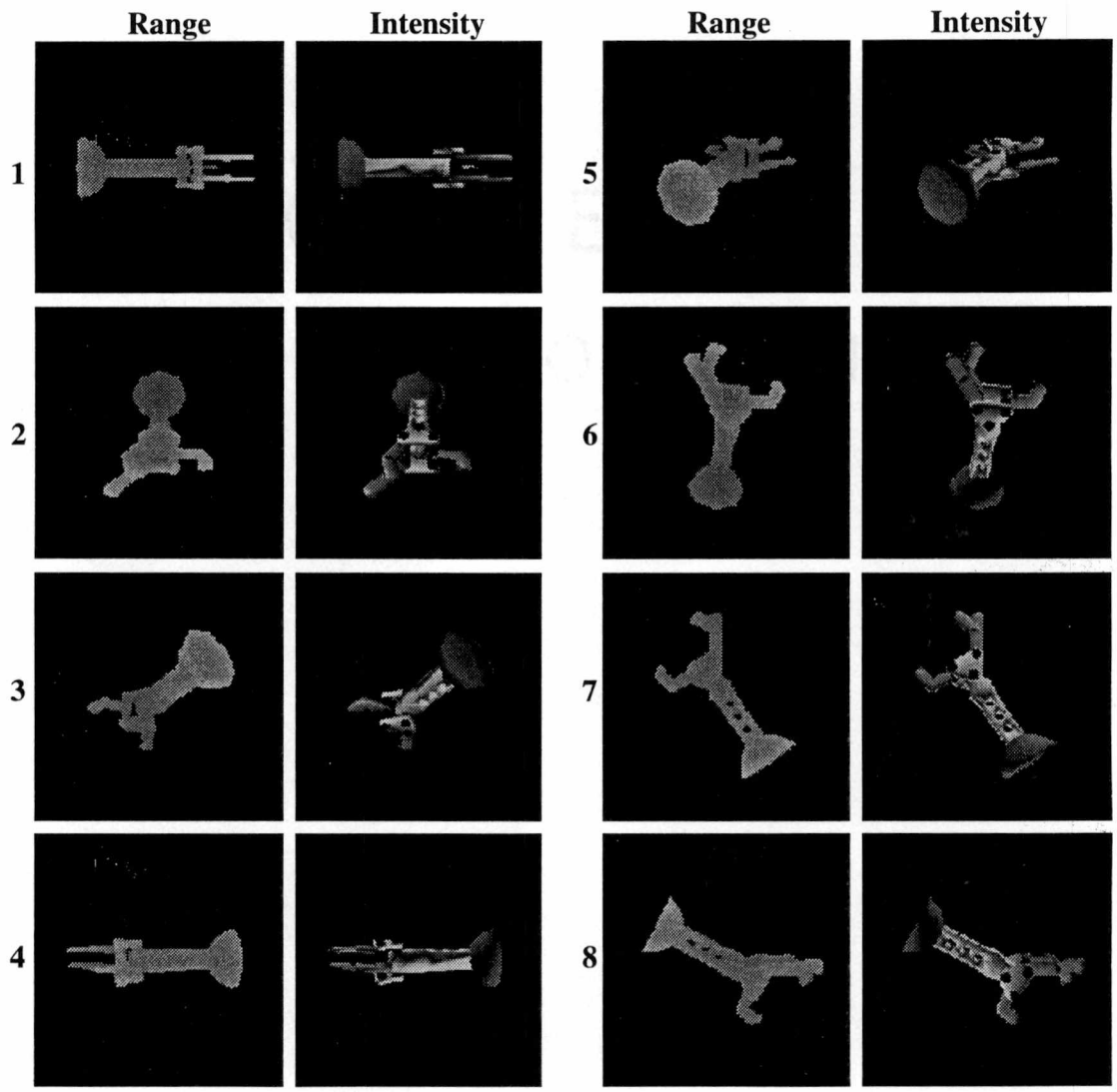


Figure 7.36: *The eight range and intensity images acquired for the hand model..*

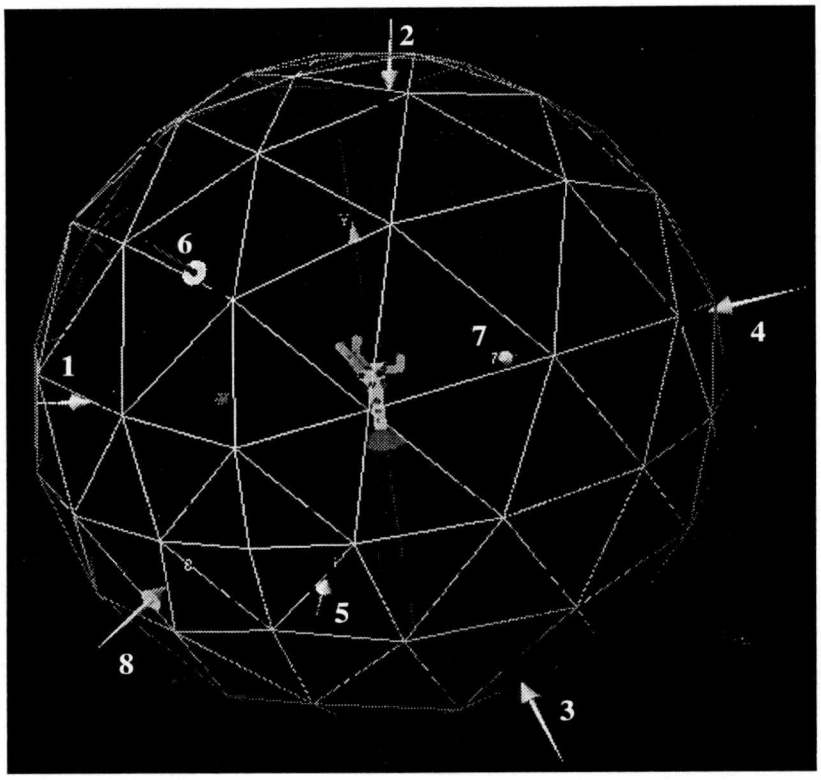


Figure 7.37: *The ideal hand model illustrated with the eight view positions.*

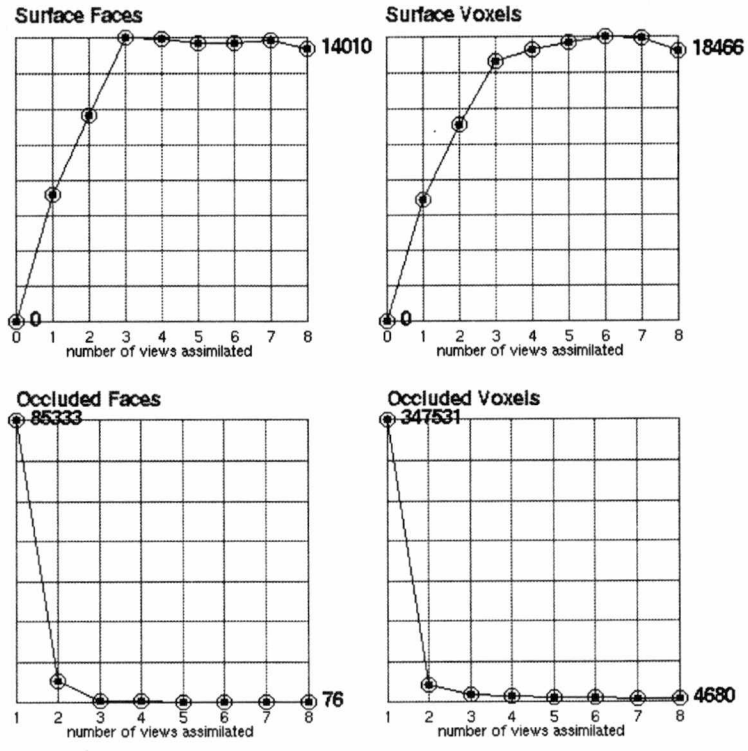


Figure 7.38: *Four graphs depicting the size of each model as reconstruction progresses.*

those voxels which appear from one range image to be on the surface of the object, but are later verified from another image as not being occupied volumes at all. If we examine a voxel rendering of the discrete surface of the reconstructed model after eight range assimilations (see Fig. 7.39), we see that many of the occlusions which we visually identified in Fig. 7.35 have been assimilated into the reconstructed prototype.

In Fig. 7.40, we see an isosurface rendering of the reconstructed hand model compared with the ideal hand model. Once more, the reconstructed model appears to compare well with the ideal model, and thus the application of extended reconstruction for this model was appropriate.

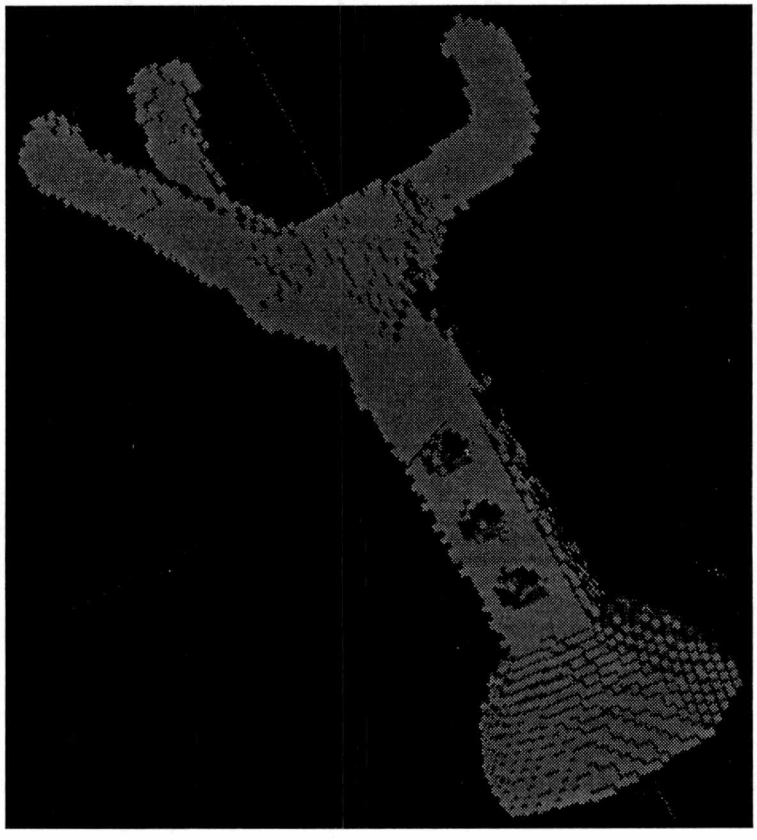


Figure 7.39: *A voxel rendering of the reconstructed hand model after the assimilation of eight range views.*

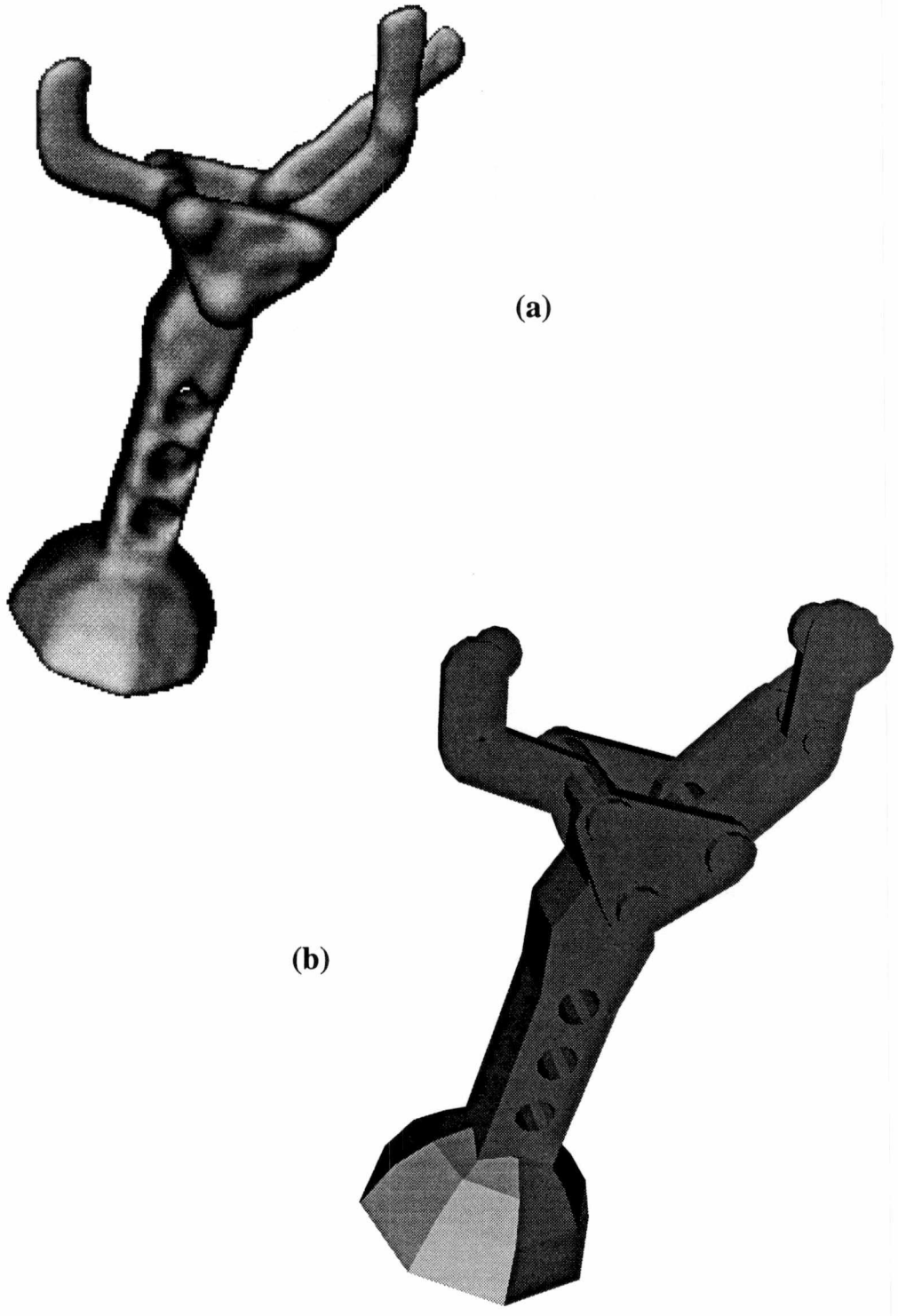


Figure 7.40: *(a) An isosurface rendering of the reconstructed hand model, compared with (b) the ideal hand model.*

CHAPTER 8

SUMMARY AND CONCLUSIONS

After reviewing the results, we feel justified in concluding that the overall performance of the best-next-view system has surpassed our original goals. The system is able to acquire range and intensity images from a specified position with respect to a given model, assimilate the range information into a 3-D bitmap model of a given size, and determine from the current status of the model the position from which an optimal amount of new range information could be acquired from the model. The system emulates the intuitive inspection and reconstruction capabilities of a human being. The models reconstructed, dependent on the object complexity and the resolution of the model, are consistently accurate and robust reproductions of the original ideal model of the object.

Figures 7.17, 7.20, 7.24, 7.28, and 7.32 each illustrate the reconstructed model of an object compared with the ideal model. The digital nature of the reconstructed voxel data and the resolutions of the models themselves prevent the reconstructed models from replicating the ideal models exactly. Of course a model of infinite (or very large) resolution would present the most accurate reproduction of any model. However, we are limited by practicality and the facilities available to us in the sizes of the models we manipulate. While the first four 256×256×256 models required in general less than an hour to reconstruct, the 400×400×400 chair model required several hours. Thus we are not only concerned with the memory occupied by the model, but also the time required to assimilate range data.

After the reconstruction of the chair model was complete, we noticed that the bottoms of the boards at the rear of the chair appeared to be eroded away. This error is due to the

relatively small thickness of the boards and a problem with the range data registration algorithm. When a thin object is viewed from the front or back, the viewed surfaces are assimilated into the prototype model (see Fig. 8.1 (b)). If the object is viewed directly parallel to the thin surface, however, when the range pixels are sampled to determine voxel occupancy, the pixels corresponding the thin surface might not be sampled because they comprise such a small part of the image (see Fig. 8.1 (c)). Because the combination procedure of the existing occupancy grid and the newly ranged occupancy grid involves a logical AND, any voxels occupied in the existing grid and not occupied in the prototype grid will not be occupied in the updated grid. Thus when ranging a thin surface with insufficient model resolution, we will see the erosion effect evident in Fig. 7.32.

Aside from this problem, the best-next-view system performed well. The application should be suitable for implementation in a practical situation, such as in the modeling of an actual three-dimensional object. Plans exist for the system to be applied to the automated modeling of a facility contaminated with radioactivity, which was the basis for the valve model. The input range images would be acquired by a Perceptron P5000 laser range scanner to be mounted on a Schilling Titan II manipulator and supervised by a dedicated

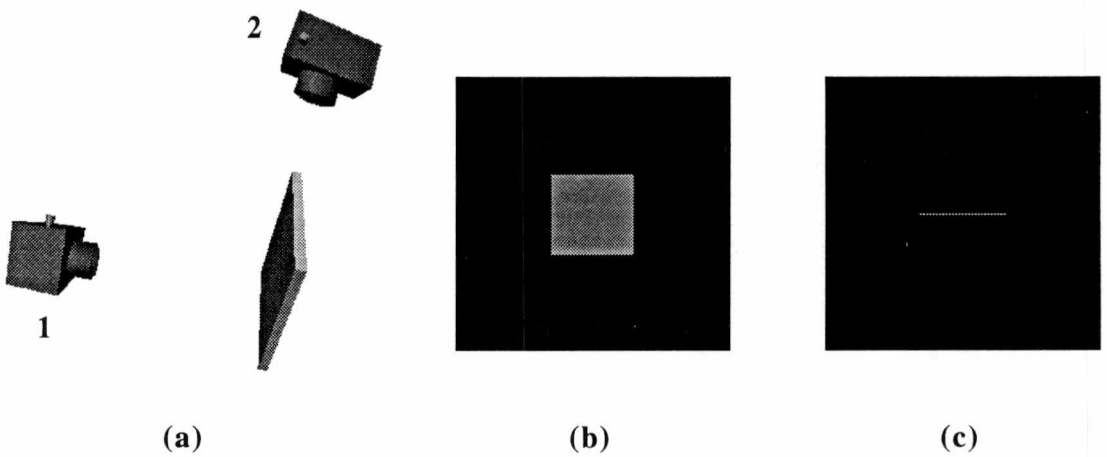


Figure 8.1: *(a) View 1 is acquired from the front of a thin surface, and View 2 is acquired parallel to the thin surface; (b) the image from View 1, and (c) the image from View 2.*

Sun SPARC 5 workstation. All elements of this proposed system exist and are functional and have only to be compiled into one working system.

Ideally, the graphical user interface of the system and the best-next view system itself should both be components of the same system. Currently, the interface, written and compiled in Java, is merely a front end for the system, written in C++ code. The system and the interface execute separately, with the interface calling the system as a native method. Because Java is an interpreted language, large data sets would bog down an application written in Java, thus making it inappropriate for a best-next-view system to be encoded in that language.

Further future work will essentially be a series of optimization projects. The matter of optimizing the C++ best-next-view system code should be a simple one; the total number of calculations with each iteration has already been minimized, but each function should be compiled inline. The most important goal for future work should be to make the system work more quickly with models of higher resolution. At the current level of technology, this is definitely possible, but faster hardware will generally be much more monetarily expensive. The current system is a compromise between hardware and software, optimized for a Silicon Graphics Indigo 2 workstation.

A definite possibility for this best-next-view system in the future will be to work with probabilistic models rather than binary models. While the current software works with models which have one bit of memory allocated for each voxel in the 3-D model, the system could easily be altered to work with models which have an entire byte of memory allocated for each voxel in the prototype model. The reader will recall that the current models were expanded to such a format for the isosurface visualization function. A probabilistic model would not only improve the accuracy of the reproduction, but the erosion effect of the registration algorithm discussed earlier in this chapter would be eliminated. Perhaps the system should work with octrees, calling on higher resolution in

the model at areas of greater physical contrast.

Optimally, the system should be limited in the range of positions which will be considered for the best-next-view position. An actual system will most likely not be able to view the object from any angle at a fixed radius from the object center. Thus a delimiter should be incorporated into the system which would be able to register range data from any radius, and inform the best-next-view system of the presence of a wall or other such object which would exclude the presence of the sensor. In this sense the system would then actually closely resemble the inspection and reconstruction capabilities of a human being in an unknown environment.

In conclusion, we should note the many separate components generated specifically for this project:

(1) The C++ class library, complete with operators and operator functions, written specifically for the purpose of handling and manipulating discrete three-dimensional models.

(2) The application which reads an IRIS Inventor format file and renders range and intensity images of the described object from a given position.

(3) The registration function which takes as arguments the existing prototype model and the most recently acquired range image, and assimilates the range information into the prototype.

(4) The best-next-view system which presents five separate methods for determining the best-next-view position from the current state of the model.

(5) The graphical user interface with which the user can call the best-next-view system, view the images acquired at each iteration, review statistics pertaining to reconstruction, or examine the ideal or reconstructed model.

(6) The application which reads in a reconstructed model file and outputs an IRIS Inventor format voxel rendering of the model for visualization.

(7) The IRIS Explorer module map which reads in a reconstructed model file and outputs an IRIS Inventor format surface rendering of the model for visualization.

In consideration of the fact that all of this work was done as the project for a Master's thesis and not a doctoral dissertation, the total amount of time and effort submitted to complete the project should exceed expectations. Most of the code written for this project is

completely original, and each component of the compiled system is fully functional. The results of the system compare well to the ideal models, and should speak for themselves.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] B. A. Abidi, "Automatic Sensor Placement," in *SPIE Intelligent Robots and Computer Vision XIV: Algorithms, Techniques, Active Vision, and Materials Handling*. SPIE—The International Society for Optical Engineering, October 1995, vol. 2588, pp. 387-98.

[2] M. A. Abidi and R. C. Gonzalez, *Data Fusion in Robotics and Machine Intelligence*, Academic Press, Inc., 1992.

[3] A. K. H. Ali, "A 2-D and 3-D Robot Path Planning Algorithm Based on Quadtree and Octree Representation of the Workspace," Master's thesis, The University of Tennessee, Knoxville, December 1987.

[4] M. Asada, "Building a 3-D World Model For A Mobile Robot From Sensory Data," *IEEE International Conference on Robotics and Automation*, 1988, pp. 918-923.

[5] J. E. Banta, Y. Zhien, X. Z. Wang, G. Zhang, M. T. Smith, and M. A. Abidi, "A 'Best-Next-View' Algorithm for Three-Dimensional Scene Reconstruction Using Range Images," in *SPIE Intelligent Robots and Computer Vision XIV: Algorithms, Techniques, Active Vision, and Materials Handling*. SPIE—The International Society for Optical Engineering, October 1995, vol. 2588, pp. 418-429.

[6] P. N. Belhumeur, "A Binocular Stereo Algorithm for Reconstructing Sloping, Creased, and Broken Contours in the Presence of Half–Occlusion," *International Conference on Computer Vision*, 1993, pp. 431-438.

[7] R. Bergevin, D. Laurendeau, and D. Poussart, "Registering Range Views of Multipart Objects," *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 1-16, January 1995.

[8] P. J. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239-256, February 1992.

[9] P. J. Besl, Sur*faces in Range Image Understanding, Sp*ringer-Verlag, New York, NY, 1986.

[10] S. M. Bhandarkar and A. Sieber, "Integrating Edge and Surface Information for Range Segmentation," *Pattern Recognition*, vol. 25, no. 9, pp. 947-962, 1992.

[11] E. Bittar, S. Lavallé, and R. Szeliski, "A method for registering overlapping range images of arbitrarily shaped surfaces for 3-D object reconstruction," in *SPIE Sensor Fusion VI*. SPIE—The International Society for Optical Engineering, 1993, vol. 2059, pp. 384-395.

[12] T. O. Binford, "Survey of Model-Based Image Analysis Systems," in *The International Journal of Robotics Research*, February 1982, vol. 1, pp. 1-3.

[13] J. Blinn, "'Where Am I? What Am I Looking At,'" *IEEE Computer Graphics & Applications*, pp. 76-81, July 1988, California Technical Institute.

[14] K. L. Boyer and A. C. Kak, "Color-Encoded Structured Light for Rapid Active Ranging," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 1, pp. 14-28, 1987.

[15] D. J. Braunegg, "A System That Recognizes World Locations with Stereo Vision," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 3, pp. 303-308, June 1993.

[16] A. Cameron and H. Durrant-Whyte, "A Bayesian Approach to Optimal Sensor Placement," *The International Journal of Robotics Research*, vol. 9, no. 5, pp. 70-88, October 1990.

[17] B. Chase, "Optical Kinematics and the Calibration of Laser Range Cameras," Master's thesis, The University of Tennessee, Knoxville, May 1997.

[18] Y. Chen and G. Medioni, "Surface Description of Complex Objects from Multiple Range Images," in *Proceedings of the 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1994, pp. 153-158.

[19] C. H. Chien, Y. B. Sim, and J. K. Aggarwal, "Generation of Volume/Surface Octree from Range Data," in *Proceedings of the 1988 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1988, pp. 254-260.

[20] C. I. Connolly, "The Determination of Next Best Views," in *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, 1985, pp. 432-435.

[21] C. K. Cowan, "Model-Based Synthesis of Sensor Location," in *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*, April 1988, vol. 2, pp. 900-905.

[22] C. K. Cowan and Peter D. Kovesi, "Automatic Sensor Placement from Vision Task Requirements," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, pp. 407-416, May 1988.

[23] C. K. Cowan and A. Bergman, "Determining the Camera and Light Source Location for a Visual Task," in *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, 1989, vol. 1, pp. 509-514.

[24] S. Dickinson, S. Stevenson, E. Amdor, J. Tsotsos, and L. Olsson, "Integrating Task-Directed Planning with Reactive Object Recognition," in *SPIE Intelligent Robots and Computer Vision XII: Algorithms and Techniques*. SPIE—The International Society for Optical Engineering, September 1993, vol. 2055, pp. 212-224.

[25] J. Ens and P. Lawrence, "An Investigation of Methods for Determining Depth from Focus," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 2, pp. 97-108, February 1993.

[26] O. D. Faugeras and M. Hebert, "The Representation, Recognition, and Positioning of 3-D Shapes from Range Data," in *Techniques for 3-D Machine Perception*, 1986, pp. 13-51.

[27] H. Gagnon, M. Soucy, R. Bergevin, and D. Laurendeau, "Registration of Multiple Range Views for Automatic 3-D Model Building," in *Proceedings of the 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 1994, pp. 581-586.

[28] L. A. Gerhardt and K. Hyun, "View Planning for Verification Using Machine Vision," in *Proceedings ICIP-94*, November 1994, vol. 3, pp. 217-220.

[29] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Addison-Wesley Publishing Company, 1992.

[30] G. D. Hager, "Task-Directed Computation of Qualitative Decisions from Sensor Data," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 14, pp. 415-429, August 1994.

[31] K. Higuchi, M. Hebert, and K. Ikeuchi, "Building 3-D Models from Unregistered Range Images" in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, May 1994, pp. 2248-2253.

[32] S. A. Hutchinson and A. C. Kak, "Planning Sensing Strategies in a Robot Work Cell with Multi-Sensor Capabilities," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 6, pp. 765-783, December 1989.

[33] Y. K. Hwang and N. Ahuja, "Gross Motion Planning—A Survey," ACM Computing *Surveys, vol. 24*, no. 3, pp. 219-291, September 1992.

[34] R. A. Jarvis, "A Perspective for Range Finding Techniques for Computer Vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-5, pp. 122-139, April 1993.

[35] H. Kim, R. C. Jain, and R. A. Volz, "Object Recognition Using Multiple Views," in *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, March 1985.

[36] J. Lavest, G. Rives, and M. Dhome, "Three-Dimensional Reconstruction by Zooming," in *IEEE Transactions on Robotics and Automation*, vol. 9, no. 2, pp. 196-207, April 1993.

[37] A. Li and G. Crebbin, "Octree Encoding of Objects from Range Images," in *Pattern Recognition*, vol. 27, no. 5, pp. 727-739, May 1994.

[38] A. D. Marshall and D. R. Roberts, "Automatically Planning the Inspection of Three-Dimensional Objects Using Stereo Vision," *SPIE Sensor Fusion and Networked Robotics VIII*, SPIE—The International Society for Optical Engineering, October 1995, vol. 2588.

[39] J. Maver and R. Bajcsy, "Occlusions as a Guide for Planning the Next View," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 417-433, May 1993.

[40] J. Maver, A. Leonardis, and F. Solina, "Planning the Next View Using the Max–Min Principle," in *Computer Analysis of Images and Patterns*, Budapest, Hungary, September 1993, pp. 543-547, Springer-Verlag.

[41] J. D. Murray and W. Van Ryper, *Encyclopedia of Graphics File Formats*, O'Reilly & Associates, First Ed., July 1994

[42] O. Monga, S. Benayoun, and O. D. Faugeras, "From Partial Derivatives of 3D Density Images to Ridge Lines," in *Proceedings of the 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 1992.

[43] A. Papoulis, *Probability, Random Variables, and Stochastic Processes,* McGraw-Hill, New York, NY, 1984.

[44] R. Pito, "A Solution to the Next Best View Problem for Automated CAD Model Acquisition of Free-form Objects Using Range Cameras," in *SPIE Modeling, Simulation, and Control Technologies for Manufacturing*. SPIE—The International Society for Optical Engineering, October 1995, vol. 2596, pp. 418-429.

[45] M. Rioux, "Laser range finder based on synchronized scanners," in *Applied Optics,* November 1984, vol. 23, no. 21, pp. 3837-3844.

[46] M. Rutishauser, M. Stricker, and M. Trobina, "Merging Range Images of Arbitrarily Shaped Objects," in *Proceedings of the 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 1994.

[47] S. W. Sedas-Gersey, "Algorithms for Automatic Sensor Placement to Acquire Complete and Accurate Information," PhD dissertation, Carnegie- Mellon University, May 1993.

[48] C. Sheckar and R. Chellappa, "Passive Ranging Using a Moving Camera," in *Journal of Robotic Systems*, vol. 9, no. 6, pp. 729-752, 1992.

[49] Silicon Graphics, Inc., "IRIS Explorer," (visualization software), October 1994, version 2.2.2, The Numerical Algorithms Group, Oxford, UK.

[50] J. C. Sluder and M. A. Abidi, "Parallel Range Data Processing: A Real Case Study," in *SPIE Intelligent Robots and Computer Vision*, November 1991, vol. 168, pp. 248-261.

[51] M. Soucy and D. Laurendeau, "Generating Non-Redundant Surface Representations of 3D Objects Using Multiple Range Views," in *Proceedings of the Tenth International Conference on Pattern Recognition*, June 1990,vol. 1, pp. 198-200.

[52] M. Soucy and D. Laurendeau, "Multi-Resolution Surface Modeling from Multiple Range Views," in *Proceedings of the 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 1992, pp. 348-353.

[53] M. Soucy and D. Laurendeau, "A General Surface Approach to the Integration of a Set of Range Views," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 4, pp. 344-358, April 1995.

[54] K. Taranabis and R. Y. Tsai, "Viewpoint planning: the visibility constraint," in *Proceedings of the Image Understanding Workshop*, May 1989, pp. 893-903.

[55] K. Taranabis, R. Y. Tsai, and P. K. Allen, "Automatic Sensor Planning for Robotic Vision Tasks," in *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, California, April 1991, pp. 76-82.

[56] K. A. Taranabis, R. Y. Tsai, and P. K. Allen, "The MVP Sensor Planning System for Robotic Vision Tasks," *IEEE Transactions on Robotics and Automation*, vol. 11, pp. 72-85, February 1995.

[57] K. A. Taranabis, P. K. Allen, and R. Y. Tsai, "A Survey of Sensor Planning in Computer Vision," *IEEE Transactions on Robotics and Automation*, vol. 11, pp. 86-104, February 1995.

[58] G. H. Tarbox and S. N. Gottschlich, "Planning for Complete Sensor Coverage in Inspection", *Computer Vision and Image Understanding*, vol. 61, pp. 84-111, January 1995.

[59] L. E. Wixson, *Gaze Selection for Visual Search*, PhD dissertation, University of Rochester, Rochester, NY, 1994.

[60] S. Xie, T. W. Calvert, and B. K. Bhattacharya, "Planning Views for the Incremental Construction of Body Models," in *Proceedings of the Eighth International Conference on Pattern Recognition*, October 1986, pp. 154-157.

[61] S. Yi, R. M. Haralick, and L. G. Shapiro, "Optimal Sensor and Light Source Positioning for Machine Vision," *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 122-137, January 1995.

[62] J. Yngvesson and I. Wallin, *The SIPP Rendering Library* (computer software available via anonymous ftp from "ftp.x.org"), Free Software Foundation, Cambridge, MA,

[63] X. Yuan, "A Mechanism of Automatic 3D Object Modeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 3, pp. 307-311, March 1995

APPENDIX

APPENDIX

CLASS DECLARATIONS

1. Class VolGrid

Class VolGrid contains the following members:

```
private:
  int Xdim;                                          // 'length'
  int Ydim;                                          // 'width'
  int Zdim;                                          // 'height'
  unsigned char* array;                              // the binary array

public:                                              // member functions
  VolGrid(int xsiz = 1,int ysiz = 1,int zsiz = 1);   // constructor
  VolGrid(const VolGrid&);                           // copy constructor
  ~VolGrid();                                        // destructor
  const int xsize(void);                             // return x size
  const int ysize(void);                             // return y size
  const int zsize(void);                             // return z size
  VolGrid& read(ifstream&);                          // read array from file
  VolGrid& read_file(char*);                         // file input
  VolGrid& write_file(char*);                        // file output
  VolGrid subgrid(int, int, int, int, int, int);     // sub-grid array
  VolGrid& insert(int, int, int, VolGrid);           // sub-grid assignment
  VolGrid& resize(int, int, int);                    // resize grid
  VolGrid& empty();                                  // completely empty grid
  VolGrid& fill();                                   // completely fill grid
  VolGrid& empty(long);                              // voxel = 0
  VolGrid& fill(long);                               // voxel = 1
  VolGrid& empty(int,int,int);                       // voxel = 0
  VolGrid& fill(int,int,int);                        // voxel = 1
  VolGrid& empty(int, int, int, int, int, int);      // clear specified voxels
  VolGrid& fill(int, int, int, int, int, int);       // fill specified voxels
  int no_error_index(const int,const int,const int); // error-free access

                                                     // member operators
  int operator () (const int,const int,const int);   // access element
  unsigned char operator () (const int index);       // direct element access
  VolGrid& operator = (const VolGrid&);              // VolGrid assignment
  VolGrid& operator = (Octree&);                     // Octree decompression
  VolGrid& operator = (unsigned char *);             // assumed size array
```

```
int operator == (const VolGrid&);          // equal test
int operator != (const VolGrid&);          // not equal test
VolGrid operator ~ (void);                 // NOT logical operator
VolGrid operator ^ (const VolGrid&);       // voxel-wise XOR
VolGrid operator & (const VolGrid&);       // voxel-wise AND
VolGrid& operator |= (const VolGrid&);     // voxel-wise |=
VolGrid& operator &= (const VolGrid&);     // voxel-wise &=
VolGrid operator | (const VolGrid&);       // voxel-wise OR
VolGrid operator + (const VolGrid&);       // voxel-wise OR
VolGrid operator - (const VolGrid&);       // "subtraction"
```

2. Class Octree

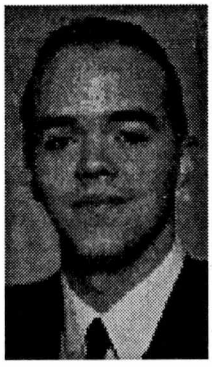
Class Octree has the following members:

```
private:
 int n;                                    // (2^n)x(2^n)x(2^n) grid
 unsigned char* array;                     // the binary array

public:                                    // member functions
 Octree(int n = 7);                        // constructor
 Octree(const Octree&);                    // copy constructor
 Octree(VolGrid&);                         // copy constructor
 ~Octree();                                // destructor
 Octree& read_file(char*);                 // file input
 Octree& write_file(char*);                // file output
 Octree& combine(VolGrid&,VolGrid&);       // two grids into one octree
 Octree& split(VolGrid&,VolGrid&);         // two grids from one octree
 int gridsize();                           // n access

                                           // member operators
 Octree& operator = (Octree&);             // Octree to Octree copy
 Octree& operator = (VolGrid&);            // VolGrid to Octree copy
 unsigned char operator () (const int index)  // direct element access
```

VITA



Joseph Banta was born in Sevierville, Tennessee in 1972. He graduated from Gatlinburg–Pittman High School in 1990, receiving recognition as a National Merit Scholar. In 1994 he graduated with honors from the University of Tennessee, Knoxville, with a Bachelor of Science in Electrical Engineering. He continued his academic career at that university seeking a Master of Science in the area of Computer Vision, specifically researching the optimal placement of range sensors during scene reconstruction. On December 22, 1995, he married Kristye Spence, a biomedical engineer also pursuing a Master's Degree.