To the Graduate Council:

I am submitting herewith a dissertation written by Christopher S. Gourley entitled **"Pattern Vector Based Reduction of Large Multimodal Data Sets for Fixed Rate Interactivity During Visualization of Multiresolution Models."** I have examined the final copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy with a major in Electrical Engineering.

M. A. Abidi

Prof. M. A. Abidi, Major Professor

We have read this dissertation
and recommend its acceptance:

Daniel B. Kyrch

Mongi A. Abidi

Ronald W. Brockett

R. J. Duhy

Accepted for the Council:

Associate Vice Chancellor and
Dean of The Graduate School

PATTERN VECTOR BASED REDUCTION OF LARGE MULTIMODAL DATA SETS FOR FIXED RATE INTERACTIVITY DURING VISUALIZATION OF MULTIRESOLUTION MODELS

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Christopher S. Gourley

December 1998

ACKNOWLEDGEMENTS

I would like to thank the many people who have contributed in some way to the completion of this dissertation. First, I would like to thank my parents, Alfred and Shirley Gourley, for their support over the years. Thanks to my advisor, Dr. Mongi A. Abidi, for his guidance throughout my program. Also I would like to thank all of my committee members, Dr. Mongi A. Abidi, Dr. Donald W. Bouldin, Dr. Rajiv V. Dubey, Dr. Daniel B. Koch, and Dr. Philip W. Smith, for their valuable inputs and suggestions. I would like to also thank Dr. Ross T. Whitaker for his advice in the data structure development for this research. A special word of thanks goes to Dr. Christophe Dumont for his daily recommendations and assistance during the research and preparation of this dissertation. And I would like to thank God "For the LORD giveth wisdom: out of his mouth cometh knowledge and understanding." This work was supported by the DOE's University Research Program in Robotics (Universities of Florida, Michigan, New Mexico, Tennessee, and Texas) under grant DOE–DE–FG02–86NE37968.

ABSTRACT

The main focus of the research presented in this dissertation is real-time visualization of large photo-realistic models created from multimodal data sets. These models are derived from range and intensity data acquired from a laser range camera along with color, thermal, or radiation data from the scene. The capability to maintain a constant display rate when dealing with these large models is desired in addition to the ability for multiple users to interact with the data. A 3D virtual reality environment is perfect for interaction with and visualization of the models created from the data sets that have been acquired. To achieve our goal, a tool for visualization consisting of both hardware and software is designed and implemented. The hardware is based around the concept of a CAVE system comprised of a large screen and several projectors. The hardware setup employed is known as the MERLIN (Multi-usER Low-cost INtegrated) visualization system. This includes a desktop SGI computer driving three VGA projectors which display onto a custom-built screen along with several VR interface devices. To maintain a constant display rate, since the number of triangles that a specific machine can draw each second is fixed, a means by which the number of triangles can be adjusted is needed. This requires both a reduction method and a multiresolution representation. The multiresolution modeling technique that is presented is a pattern vector based technique known as POLYMUR (POLYgon MUltimodal Reduction) which is capable of handling the multimodal data sets. This method outputs a multiresolution file which can be used to automatically select the proper resolution needed to maintain the user's desired frame rate when interacting with the model and fill in the details when the model is stationary.

Contents

1. Introduction **1**
- 1.1 Research Problem . . . 1
- 1.2 Research Methodology and Objectives . . . 3
- 1.3 Unique Contributions . . . 6
- 1.4 Organizational Overview . . . 7

2. Multi-user Large Screen Display **9**
- 2.1 CAVE History . . . 10
- 2.2 State-of-the-Art . . . 12
  - 2.2.1 Electronic Visualization Laboratory (NCSA) . . . 13
  - 2.2.2 Iowa Center for Emerging Manufacturing Technology . . . 13
  - 2.2.3 Stanford Computer Science . . . 14
  - 2.2.4 Massachusetts Institute of Technology . . . 16
  - 2.2.5 Other Large Screen Deployments . . . 16
- 2.3 VR Hardware Interface for MERLIN . . . 17
- 2.4 Design of Our CAVE Setup . . . 22
  - 2.4.1 The Screen . . . 22
  - 2.4.2 The Projectors . . . 25
  - 2.4.3 The Graphics Engine . . . 26
  - 2.4.4 The Desktop CAVE . . . 28
  - 2.4.5 Future CAVE Hardware Consideration . . . 31
- 2.5 User Interface Design for Large Screen Display of Models . . . 31
  - 2.5.1 Viewing Large Images . . . 34
  - 2.5.2 360° Image Viewing . . . 36
  - 2.5.3 3D Model Viewing and Manipulation . . . 37
- 2.6 Conclusions . . . 37

3. Multiresolution Level-of-Detail Review **41**
- 3.1 Terminology . . . 41
- 3.2 Overview of Polygon Reduction . . . 47
- 3.3 Height Fields . . . 50
- 3.4 Manifold Methods . . . 52
  - 3.4.1 Manifold Refinement . . . 52
  - 3.4.2 Manifold Decimation . . . 52
  - 3.4.3 Coplanar Facet Merging . . . 52
  - 3.4.4 Vertex Decimation . . . 54
  - 3.4.5 Edge Contraction and Mesh Optimization . . . 55
  - 3.4.6 Volume Methods . . . 58
  - 3.4.7 Simplification Envelopes . . . 59
  - 3.4.8 Wavelet Surfaces . . . 61
  - 3.4.9 Others . . . 61
- 3.5 Non-Manifold Methods . . . 62
  - 3.5.1 Vertex Clustering . . . 62
- 3.6 Conclusions . . . 63

4. Pattern Vector Based Mesh Reduction and Multiresolution Representation **66**
4.1 Goal . . . 66
4.2 Reduction Methodology . . . 67
4.2.1 Pattern Vectors . . . 69
4.2.2 Data Structures and Representation . . . 72
4.3 Reduction Implementation . . . 75
4.3.1 Error Measurement . . . 78
4.4 Conclusions . . . 82

5. Experimental Results **85**
5.1 Feature Based Edge Length Calculation . . . 85
5.2 Pattern Vector Based Mesh Reduction Implementation Results . . . 87
5.2.1 Synthetic Data Model Results . . . 90
5.2.2 Perceptron Range Data with Segmentation Information . . . 94
5.2.3 Perceptron Range Data with Color Texture Mapped Image . . . 99
5.2.4 Coleman Range Data with Confidence . . . 104
5.2.5 Digital Elevation Map . . . 108
5.2.6 Fused Range Data Sets . . . 111
5.3 Error Analysis . . . 114
5.4 Automatic Resolution Selection for Constant-Rate Interactivity . . . 120
5.5 CAVE System with Automatic Reduction . . . 123
5.6 Conclusions . . . 123

6. Conclusions and Future Work **126**
6.1 Future Work . . . 128

BIBLIOGRAPHY **131**

APPENDICES **140**

A. Theory and Background **141**
A.1 Basic 3D Transforms . . . 141
A.2 Camera Model . . . 143
A.3 Texture Mapping . . . 144
A.4 Calculating Surface Normals for Improved Visual Quality . . . 147
A.5 Neural Networks . . . 149
A.5.1 The Artificial Neuron . . . 150
A.5.2 Artificial neural network . . . 152
A.5.3 Learning . . . 153
A.5.4 Recall . . . 153
A.6 Range Scanning . . . 153
A.7 Simulated Range Scanning . . . 155

B. Virtual Reality **157**
B.1 Virtual Reality Overview . . . 157
B.2 Virtual Reality Hardware . . . 159
B.2.1 Video Interface . . . 160
B.2.2 Audio Interface . . . 163
B.2.3 Haptic Interface . . . 164
B.2.4 Position and Tracking Interface . . . 165
B.2.5 Other Interface Hardware . . . 166

B.3 Applications of Virtual Reality 166
B.4 Glossary of Acronyms 169
B.5 Glossary of Terms 170

VITA **172**

List of Figures

1.1 *Flowchart of the IRIS overall long-term goal, the creation of models from range images and the visualization of and interaction with those models. The main focus of this research is in the display of the data.* 2
1.2 *Overview of the visualization system.* 5
2.1 *Cave drawings discovered in December of 1994 at Vallon-Pont-d'Arc in the Ardèche region of France. These drawings are thought to be the oldest known making this the first CAVE (image courtesy of http://www.culture.fr).* 9
2.2 *Four projector CAVE in place at EVL (image courtesy of EVL, University of Illinois, Chicago).* 13
2.3 *Four projector CAVE in place at Iowa State University (image courtesy of Iowa State University).* 14
2.4 *The Responsive Workbench, a large screen table display (image courtesy of Stanford University).* 15
2.5 *Connections of all hardware used for user interaction with the data.* 18
2.6 *VR hardware used including (1) a Virtual Technologies'* CyberGlove*, (2) a Polhemus Fastrak, and (3) a Spacetec Spaceball.* 19
2.7 *Diagram showing the acquisition of data from the DataGlove and Polhemus tracker and the communications used to send the data to the user interface.* 20
2.8 *Model of the reconfigurable screen shown with a radius for 120° circular view.* . 23
2.9 *Two other possible screen configurations.* 23
2.10 *The six frames hinged together to form the large reconfigurable screen.* 24
2.11 *Polaroid Polaview 110 LCD projector.* 25
2.12 *Silicon Graphics Indigo2 Maximum Impact with Impact Channel Option.* 27
2.13 *Layout of the room containing the desktop CAVE consisting of three projectors driven by an SGI MaxImpact and projecting onto a custom-built screen.* 29
2.14 *Fish-eye view of the room housing the CAVE showing the current setup.* 30
2.15 *View of the ICO frame buffer showing three views of same scene to create one seamless view to be displayed with the projectors along with one view for the user interface. The model shown was created from several sets of range data supplied by Oak Ridge National Laboratories acquired by a Coleman laser range scanner. The model is texture mapped with color coded quality values returned from the scanner.* 32
2.16 *Three cameras in one of many possible configurations all looking at the same scene to create one continuous view.* 33
2.17 *High resolution image displayed on the CAVE.* 34
2.18 *View of the ICO frame buffer showing three views of same image to create one seamless view to be displayed with the projectors along with one view for the user interface.* 35
2.19 *High resolution image displayed on the CAVE.* 36
2.20 *Spherical image of the environment texture mapped onto a sphere.* 37
2.21 *Setup configured as 3 flat screens giving a 160° field-of-view displaying a model created from the laser range data supplied by ORNL.* 38
2.22 *Color difference visible at the seams of the screen where edge matching is performed.* 39
3.1 *(1) Triangle mesh created from a range image of a flat surface composed of 4802 polygons, and (2) the reduced geometry of the flat surface composed of only 2 triangles.* 42
3.2 *Terminology used throughout this chapter.* 43

3.3 *Simplices and their simplicial neighborhoods.* 45
3.4 *Different level-of-detail models created using images pyramids of a synthetic range image of a plane with resolutions of (1) 90x90 pixels, (2) 60x60 pixels, and (3) 30x30 pixels.* 46
3.5 *Image pyramid creation.* 51
3.6 *Coplanar facet merging.* 53
3.7 *Vertex decimation.* 54
3.8 *Edge contraction.* 56
3.9 *Simplification envelopes concept performed on a two-dimensional model.* 60
3.10 *Vertex clustering on a two-dimensional model.* 63
4.1 *Dendrogram tree structure used to represent the multiresolution mesh created from the edge collapse reduction.* 68
4.2 *Mapping of vectors into the feature space to calculate edge lengths used in mesh reduction. In this case* $\mathbb{R}^2$ *is mapped to* $\mathbb{R}^3$. 71
4.3 *Data structure diagram.* 74
4.4 *Edges and faces marked for removal (darkly colored) and update (lightly colored).* 76
4.5 *Resulting edges and faces after removal and update.* 77
4.6 *Flowchart of the reduction method.* 79
4.7 *Calculation of error created by removing faces.* 80
4.8 *Model shown with its associated bounding box used in percentage error calculation.* 81
4.9 *(1) Original model created from a synthetic range scan, (2) model after 90.2% reduction, and (3) the difference from the two projective views.* 82
4.10 *Thresholded version of the visual error image showing the areas of change between the original and reduced models.* 83
5.1 *Color coded edges lengths based on weights using (1) geometry only, (2) normal only, (3) geometry and normal, (4) boundary only, (5) curvature only, and (6) all vectors weighted equally.* 86
5.2 *Typical vector weighting with the geometry weight = 1, normal weight = 0.5, boundary weight = 2.0, and curvature weight = 1.5.* 88
5.3 *Amount of time versus the number of edges collapsed for the various models.* . . 89
5.4 *256x256 range image resulting in a model containing 160,698 edges and requiring 305MB of memory for reduction.* 90
5.5 *(1a) Initial model from synthetic range data with 39,966 faces, (1b) wire-frame of the initial model, (2a) model after 60.3% reduction, and (2b) wire-frame of the 60.3% reduced model.* 91
5.6 *(1a) Model after 90.2% reduction, (1b) wire-frame of the 90.2% reduced model, (2a) model after 97.8% reduction, and (2b) wire-frame of the 97.8% reduced model.* 92
5.7 *(1a) Model after 99.6% reduction, (1b) wire-frame of the 99.6% reduced model, (2a) model after 99.9% reduction, (2b) wire-frame of the 99.9% reduced model.* . 93
5.8 *Maximum calculated error versus the number of triangles for the model created from a synthetic range image.* 94
5.9 *(1) 256x256 range image taken with the Perceptron laser range camera only, (2) registered intensity image taken with the Perceptron laser range camera, and (3) segmentation of the range image.* 95
5.10 *(1a) Initial Perceptron model with 116,544 faces, (1b) wire-frame of the initial Perceptron model, (2a) Perceptron model after 72.6% reduction, and (2b) wire-frame of the 72.6% reduced model.* 96
5.11 *(1a) Perceptron model after 91.5% reduction, (1b) wire-frame of the 91.5% reduced model, (2a) Perceptron model after 96.1% reduction, and (2b) wire-frame of the 96.1% reduced model.* 97

5.12 *(1a) Zoomed view of the initial Perceptron model showing the pipe in the middle of the scene, (1b) wire-frame of the initial model, (2a) zoomed view after 62.5% reduction, and (2b) wire-frame of the 62.5% reduced model.* 98
5.13 *Maximum calculated error versus the number of triangles for the real range image with segmentation information.* 99
5.14 *(1) 768x511 range image taken with the Perceptron laser range camera, and (2) a registered color image.* 100
5.15 *(1a) Initial color model with 179,984 faces, (1b) wire-frame of the initial color model, (2a) color model after 60.4% reduction, and (2b) wire-frame of the 60.4% reduced model.* 101
5.16 *(1a) Color model after 80.3% reduction, (1b) wire-frame of the 80.3% reduced model, (2a) color model after 94.2% reduction, and (2b) wire-frame of the 94.2% reduced model.* 102
5.17 *(1a) Color model after 99.0% reduction without using color information, and (1b) wire-frame of the 99.0% reduced model.* 103
5.18 *Maximum calculated error versus the number of triangles for the full 3D model created from 12 range images.* 104
5.19 *335x181 range image taken with the Coleman laser range camera.* 105
5.20 *(1a) Initial coleman model with 118,996 faces, (1b) wire-frame of the initial coleman model, (2a) coleman model after 70.4% reduction, and (2b) wire-frame of the 70.4% reduced model.* 106
5.21 *(1a) Coleman model after 90.2% reduction, (1b) wire-frame of the 90.2% reduced model, (2a) coleman model after 94.7% reduction, and (2b) wire-frame of the 94.7% reduced model.* 107
5.22 *Maximum calculated error versus the number of triangles for the full 3D model created from 12 range images.* 108
5.23 *360x440 digital elevation map of Southern Florida.* 109
5.24 *(1a) Initial model created from DEM with 100,000 faces, (1b) wire-frame of the initial model, (2a) model after 77.0% reduction, (2b) wire-frame of the 77.0% reduced model, (3a) model after 96.0% reduction, and (3b) wire-frame of the 96.0% reduced model.* 110
5.25 *Maximum calculated error versus the number of triangles for the DEM model.* . . 111
5.26 *(1a) Initial mug model with 38,268 faces, (1b) wire-frame of the initial mug model, (2a) mug model after 89.9% reduction, and (2b) wire-frame of the 89.9% reduced model.* 112
5.27 *(1a) Mug model after 99.1% reduction, (1b) wire-frame of the 99.1% reduced model, (2a) mug model after 99.8% reduction, and (2b) wire-frame of the 99.8% reduced model.* 113
5.28 *Maximum calculated error versus the number of triangles for the full 3D model created from 12 range images.* 114
5.29 *Maximum calculated error for all models versus the number percent reduction.* . 115
5.30 *(1) Visual error between the initial model created from synthetic range data and the 97.8% reduced model, and (2) visual error image thresholded.* 117
5.31 *(1) Visual error between the initial model created from Perceptron range data and the 96.1% reduced model, and (2) visual error image thresholded.* 117
5.32 *(1) Visual error between the initial model created from Perceptron range data and color imagery and the 94.2% reduced model, and (2) visual error image thresholded.* 118
5.33 *(1) Visual error between the initial model created from Coleman range data and the 94.7% reduced model, and (2) visual error image thresholded.* 118

5.34 *(1) Visual error between the initial model created from DEM data and the 96.0% reduced model, and (2) visual error image thresholded.* 119
5.35 *(1) Visual error between the initial model created from multiple range data sets and the 99.1% reduced model, and (2) visual error image thresholded.* 120
5.36 *Control loop to calculate the needed resolution to maintain a constant display rate.* 121
5.37 *Automatic resolution selection on performed using two separate machines: (1) the original model containing 38,268 triangles, (2) the resolution selected on a faster machine to maintain 60 frames per second (3,717 triangles), and (3) the resolution selected on a slower machine to maintain the same frame-rate (1,200 triangles).* . . 122
5.38 *Automatic resolution selection on performed using the CAVE display system: (1) the original model containing 38,268 triangles, (2) the resolution selected to maintain 15 frames-per-second, and (3) the resolution selected to maintain 30 frames-per-second.* 124
A.1 *Pinhole camera model with a focal length of λ for a right-handed coordinate system.* 143
A.2 *Texture of a brick mapped onto a cube.* 144
A.3 *Texture coordinates shown using a repeating texture of wood paneling.* 145
A.4 *Registered intensity image texture mapped onto a triangle mesh created from a range image.* 146
A.5 *Faceted surface shown on left versus a smoothed surface on right created using true point normals.* 147
A.6 *Normal calculations for points based on the normals from faceted polygon data.* 148
A.7 *Textured polygons shown (1) with default normals and (2) with calculated true normals.* 149
A.8 *Artificial neuron.* 151
A.9 *A fully connected feedforward artificial neural network with 4 inputs, 5 hidden nodes, and 3 outputs.* 152
A.10 *An orthogonal-axis scanner which casts a ray to the nearest object from its starting point while pivoting about its x and y axes.* 154
A.11 *(1) Simulated orthogonal-axis scanner user interface and (2) an output range image from the simulated scanner.* 156
B.1 *Breakdown of HMI hardware from VR technology.* 160
B.2 *HMD comparison.* 162
B.3 *BOOM.* 164

CHAPTER 1

Introduction

1.1 Research Problem

Creation of photo-realistic three-dimensional (3D) models has recently come to the forefront of computer vision with the advent of machines capable of producing and displaying high resolution models. One issue facing virtual reality (VR) development in general has been the rapid creation of these models. It has been stated that creating a model of one room requires the same effort as writing several thousand lines of code [2]. Everyone is interested in using these models to immerse a user in a virtual world, including real-estate agents, car manufacturers, the entertainment industry, the military, and research scientists. Recent efforts at the Department of Energy (DOE) have also been focusing on such modeling in conjunction with the dismantlement of old, hazardous facilities. The contents of many of these are unknown, therefore, before sending a person or robot into these unknown areas to begin disassembly, a model of the contents is needed to better form a plan of action in order to minimize the amount of nuclear exposure. Along these lines, the research that the Imaging, Robotics, and Intelligent Systems (IRIS) laboratory at the University of Tennessee (UT) performs deals with the creation and visualization of photo-realistic 3D models created from range and intensity data acquired from a laser range camera which can be sent into these unknown facilities to map them. Along with the range images, other data from the scene is also available which includes color, thermal, or radiation data. The overall long-term plan, which encompasses more than just the research presented in this dissertation, involves taking multiple range images from various points-of-view along with data from other sensors, combining them, and creating photo-realistic models which presents the information in a useful and meaningful manner. This involves determining the next best sensor pose [117], fusing different sets of range data [27], fusing range data with intensity data [28], as well as interpreting the data sets using segmentation [11] and object recognition [64]. Furthermore, this research in particular seeks

to focus on developing methods to quickly and efficiently manipulate and view the data in real-time. The data sets that we are dealing with are very large and require very high speed graphics hardware to handle the display of the data. A typical reconstructed scene may consist of millions of triangles for the model. The model would contain pipes, valves, barrels, walls, floors, and other objects. The details of the objects in the scene are needed because we may be looking for one small radioactive barrel in the corner of a large room. But on the other hand, with a huge number of triangles used to model the scene, it is not possible to manipulate or view the data in an efficient manner. Therefore, this research focuses on developing a method which can accomplish the task of displaying the data in real-time and still keep the high resolution needed. Figure *1.1* gives an overview of the entire research objective for the lab. This begins with data acquisition, proceeds through all the data processing, and ends with the final goal, the display of a complete, 3D, texture-mapped high level model of the environment.

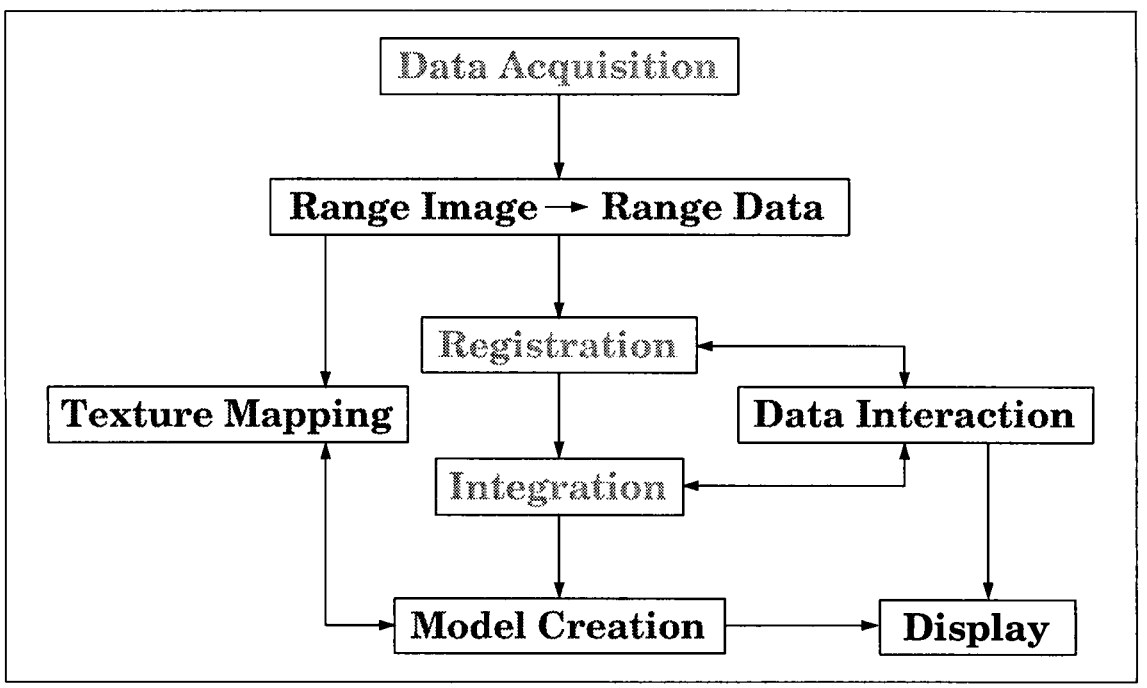


Figure 1.1: *Flowchart of the IRIS overall long-term goal, the creation of models from range images and the visualization of and interaction with those models. The main focus of this research is in the display of the data.*

1.2 Research Methodology and Objectives

The main focus of the research presented in this dissertation is *real-time visualization of large photo-realistic models* and the issues that arise while performing the visualization. This dissertation does not address all the issues of the long-term goal presented in the previous section. For example, it does not address how the range data is acquired or what sensor is used. Data conversion, model creation, and texture-mapping are touched upon, but the major focus of the present research is the interaction with the 3D models created from the data and the real-time display of those models for multiple users. The capability to maintain a constant display rate when dealing with these large models is necessary from a human factors stand-point allowing the user to easily manipulate large data sets. Another motivation, to quickly interact with the models, is because of the monetary cost associated with operations in a hazardous environments. Therefore, we have chosen to keep the interactivity level high by maintaining a constant display rate for the models. For interaction, standard human-computer interaction methods, such as the keyboard and mouse, become cumbersome because they were designed for a flat two dimensional screen. A virtual reality environment, however, is three dimensional, making the tools developed for VR perfect for interaction with and visualization of the models created from the data sets that have been acquired. To achieve our goal, a tool for visualization consisting of both hardware and software is designed and implemented. The hardware is based around the concept of a CAVE system comprised of a large screen and several projectors. The hardware setup employed is known as the MERLIN (Multi-usER Low-cost INtegrated) visualization system. This includes a desktop SGI computer, three VGA projectors, a custom-built screen, and several VR interface devices. To maintain a constant display rate, since the number of triangles that a specific machine can draw each second is fixed, a means by which the number of triangles can be adjusted is needed. This requires both a reduction method and a multiresolution representation. The multiresolution modeling technique that is presented is known as POLYMUR (POLYgon MUltimodal Reduction) and is capable of handling the multimodal data sets. On the surface this hardware and software tool is easy-to-use, while hiding the underlying complexities from the user. With this in mind, this research specifi-

cally addresses the following items:

- create texture-mapped models from registered pairs of range and intensity images,
- interact with the models using VR hardware,
- build a large screen, multiple user, interactive CAVE system,
- develop a method to create multiresolution models, and
- the real-time display of the models created.

The first item is mainly just implementation. Little research is required to complete the task. For the second task, equipment developed for virtual reality allows us to maximize the visualization of and interaction with models created from multimodal data sets. An object-oriented philosophy [113] is used in programming the system. This means that the system is comprised of several low-level modules and allows simple interaction with the user at higher levels. All low-level communication, complexities, and algorithms are transparent to the end user. Also, by providing high-level access to these modules, integration into the final overall system is easier. Therefore, by utilizing VR hardware and object-oriented programming, we are able to translate low-level complexity into a simple, easy to use, high-level tool for visualization. A flowchart of the system's use is given in Figure *1.2*. This shows the various types of range data coming into the system and being converted into the 3D data class. The 3D models along with standard images and 360° bubble images can also be displayed onto a large screen system.

To achieve the multi-user portion of our goal, we borrow from the hardware available for the VR world. The hardware that will be used for this project includes a Virtual Technologies' *Cyber*Glove, a Polhemus Fastrak, a Spacetec Spaceball and a custom-built CAVE, a large surround screen, projected visualization system. These devices are connected to a SGI Maximum Impact with an Impact Channel Option. This machine can drive four displays directly for the CAVE projectors. The glove and tracker are to be used for manipulation of range data in the virtual environment (VE), the spaceball for navigation in the VE, and the CAVE for display of the VE.

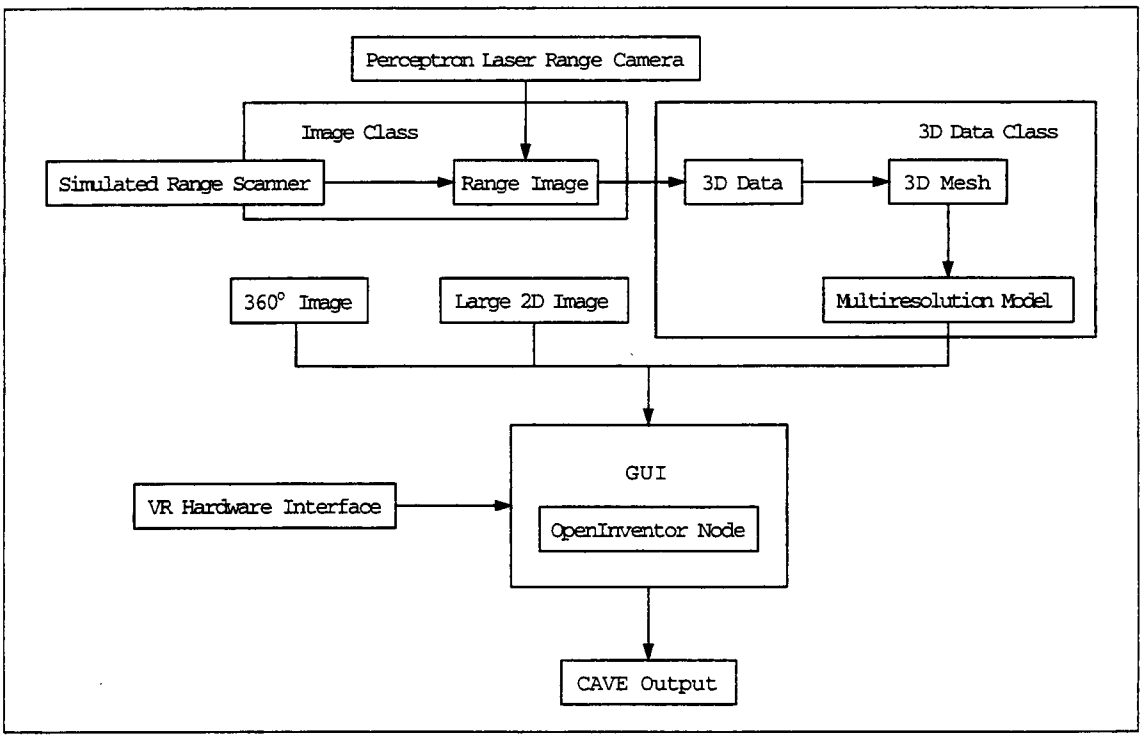


Figure 1.2: *Overview of the visualization system.*

The majority of the research, however, is in the real-time portion of displaying the data sets. To achieve constant frame-rates, the number of polygons in the models must be reduced. Therefore, a new mesh reduction method based on pattern or feature vectors is created which can handle the multimodal data with which we are dealing. Also, the displayed models will be stored in a multiresolution data format. A multiresolution model gives increased performance in that higher update rates can be achieved by using simpler models with less triangles while moving the object, and details can be shown when the model is stationary.

1.3 Unique Contributions

This dissertation sets forth three unique contributions that have been accomplished during the development of this visualization system for multimodal 3D data. These include:

- a low-cost, large screen display CAVE system, MERLIN,
- a new pattern vector based polygon mesh reduction technique, POLYMUR, and
- a new dendrogram binary tree multiresolution representation.

The CAVE system that has been designed and built is unique in that it is a low-cost system built from off-the-shelf hardware components. The setup is controlled from a desktop computer driving multimedia projectors. All previously built CAVE systems depend upon one or more large, expensive rack mounted computers. We trade off some performance for price in going with a smaller computer, but the machine we have chosen is capable of driving up to four projectors and has hardware texture mapping.

The multiresolution modeling technique that is presented is unique in that, in contrast to other reduction methods, we take a multimodal approach. Most current methods attempt to reduce the number of polygons in a mesh by using local geometry information. We apply a pattern recognition approach to merge the vertices based on features they possess. The first step in creating the multiresolution model is choosing a polygon reduction methodology. We are using a decimation methodology since we begin with a large laser range data set. Several algorithms

have been introduced to reduce polygon meshes, as is discussed in Chapter *3*. Edge contraction methods, however, are very favorable for the creation of multiresolution meshes. Here a new technique is developed to be applied to the laser data based on the edge contraction methodology. To incorporate the multimodal aspect of the data, we have chosen a pattern vector based approach to determine the similarities in the data while performing the reduction. We start with an initial mesh created from the laser range image. From here, we create a feature space based on the vertices of the initial model. The feature space allows the reduction of the model to be based on more than just geometry and takes into account such attributes as surface normals, surface curvature, color, and boundary information. The feature space can be easily extended to include other desired components such as thermal or radiation data. Edge lengths are calculated using the feature space using the connections from the initial network of edges between the vertices. The edge with the shortest length is then contracted into a new vertex and the surrounding faces and edges adjusted accordingly. This approach is applied iteratively, contracting the shortest edge and updating neighboring edge lengths at each iteration, until the desired reduction is achieved.

The reduced mesh is stored as a series of edge collapses/splits forming a multiresolution mesh from which varying resolutions can be extracted based on the end user's needs. A common data format, Open Inventor, is used for storing and displaying the data by subclassing a shape node. This allows cross-platform compatibility. The representation contains a highly detailed model along with a dendrogram or binary tree which contains the collapse/split order of the edges. This representation can quickly switch from one resolution to another. Frame rate is maintained on any machine by automatically adjusting the model to the appropriate resolution based on the user's desired interactivity.

1.4 Organizational Overview

This dissertation is arranged as follows. The CAVE concept and history is first discussed in addition to the hardware associated with this research. The issues associated with the design and construction of a CAVE are then covered along with the initial results of our design given the constraints of our system. Next, a literature review of the state-of-the-art methods for mesh reduction

is given. The multiresolution model representation for use in increasing the frame rate in visualization is then discussed as well as the implementation of the pattern vector based mesh reduction technique. The experimental results are then presented with several different examples given. The final chapter gives conclusions and some areas of future research. An appendix covering some background information and another with a review of virtual reality are also provided.

CHAPTER 2

Multi-user Large Screen Display

Cave drawings were first used thousands of years ago to visualize stories (see Figure *2.1*). Now we have the 21^{st} century equivalent of those cave drawings. Through the use of virtual reality hardware and techniques we are again drawing on walls to visualize 3D data (in our case, laser range images). The word CAVE is a recursive acronym standing for *Cave Automatic Virtual Environment*. The word CAVE suggests the appearance of this system used for visualization, but the name was actually chosen as an allusion to the Allegory of the Cave in Plato's *Republic* where ideas of perception, reality, and illusion were explored [21]. (CAVE^{TM}, Computer-Assisted



Figure 2.1: *Cave drawing "Lions and Rhinoceroses with a few red dots" discovered in December of 1994 at Grotte Chauvet, Vallon-Pont-d'Arc, Ardèche, France, and photographed by Jean Clottes. These drawings are thought to be the oldest known making this the first CAVE. (Available from http://www.culture.fr/culture/arcnat/chauvet/en/gvpda-d.htm. Accessed November 19, 1998)*

Virtual Environment, has been trademarked.) The CAVE was designed to be a tool for scientific visualization, to help match VR to real tasks and create a practical VR system. In a CAVE, the user is immersed in a virtual environment through the use of projectors displaying onto the walls. Some CAVE's also project onto the floor. This type of immersion does not require the user to "suit up" and multiple users can interact in the same VE. These aspects of the CAVE make it ideal for data visualization. This chapter deals with ideas behind the development of the CAVE and covers the current state-of-the-art system deployments. The details of the implementation of our CAVE setup are then discussed followed by initial results obtained.

2.1 CAVE History

The present day, state-of-the-art CAVE may be new, but the ideas behind it and its creation have been around longer. These ideas where first presented by Ivan Sutherland in 1965.

> The ultimate display would, of course, be a room within which a computer can control the existence of matter. ... With appropriate programming such a display could literally be the Wonderland into which Alice walked. [100]

He later built the first head-mounted display (HMD) in 1968 [101]. Working under Sutherland, Jim Clark built an HMD in 1974 and went on to develop the first geometry engine for computer graphics at Stanford. Clark took these ideas and founded Silicon Graphics, Inc., (SGI) in 1982, the first computer company specializing in high performance graphics hardware. These computers currently are almost exclusively used to drive the graphics in all CAVE's due to the geometric complexity and update rate required. In 1985, the United States Air Force built the SuperCockpit simulator. This was the first device to combine an HMD, data glove, speech recognition, 3D audio, and computer graphics. VPL Research was founded in 1986 making data gloves commercially available. In 1989, StereoGraphics was founded making stereo displays possible with their CrystalEyes shutter glasses. Also in 1989, trade shows displayed the first HMD's and datagloves. Hardware texture mapping became a reality in 1990 with the advent of VGX graphics from SGI. More VR hardware companies were formed in 1991, including Fakespace, Virtual Technologies,

and SpaceBall. The release of SGI's Reality Engine graphics in 1992 made it possible for high end graphics such as the CAVE to be implemented. Finally, the first CAVE was built in 1992 by the Electronic Visualization Laboratory (EVL) at the University of Illinois at Chicago and displayed at SIGGRAPH '92. Results from research at EVL had shown that VR technology still needed development [21, 19, 20]. Here, a CAVE was built utilizing state-of-the-art equipment. The CAVE consisted of a 27 cubic meter room in which stereo images are displayed on the 3-by-3 meter structure with rear-projected walls and a front projected floor. The user wares a pair of shutter glasses in the room to get the feel of immersion. These glasses also had an attached tracking device to determine the user's location in the room. The hardware for this unique setup cost $600,000 and consisted of:

- Four SGI Crimson VGX workstations with 256MB RAM,
- One SGI Personal Iris (master controller),
- Crystal Eyes stereo glasses,
- MIDI synthesizers for 3D sound, echoes, and Doppler shifts,
- Eight speakers, one in each corner,
- Flock of Birds tracker,
- ScramNet optical-fiber network,
- Four Electrohome Marquee 8000 projectors capable of stereo display at a resolution of 1280x512 and update rates of 120 Hz, and
- Tracking wand for manipulation.

The result is a non-intrusive, multi-user VR environment with the resolution associated with a binocular omni-orientational monitor (BOOM) but not the limited movements. At the time this hardware provided four times the resolution available on an HMD. Unfortunately for multiple users, only one perspective could shown on the screens at one time. EVL was joined in 1992

by the National Center for Supercomputing Applications' (NCSA) Virtual Environment Group (VEG) at the University of Illinois, Urbana-Champaign, to further develop the CAVE [67]. The current hardware setup has been upgraded with:

- Two SGI Onyx RE with eight processors, 1 GB of RAM, and two graphics pipes each,
- Two SGI Indies with four speakers for 3D sound, and
- The NCSA POWERCHALLENGEarray consisting of eight SGI's with a total of 84 CPU's for data processing.

CAVE's are now being sold by Pyramid Systems of Southfield, MI, and several CAVES have been deployed. This includes CAVE's at Caterpillar [40], General Motors, EDS Detroit VR Center, and FMC in San Francisco [26]. Also, CAVE's are located at ARPA Enterprise in Arlington, VA, Argonne National Laboratory in Argonne, IL, and the Iowa Center for Emerging Manufacturing Technology at Iowa State [9]. At SIGGRAPH '96 Pyramid Systems unveiled the Immersadesk, a one wall portable CAVE.

In 1994, the German National Research Center for Computer Science created the Responsive Workbench [60, 12]. This device is very similar to the cave except that instead of displaying the virtual environment onto the walls around the users, they are displayed on a table top in front of the users. In 1996, SGI revealed there own version of the CAVE, the SGI Reality Center Visionarium [53]. The same year SGI also released the first desktop computer with the graphics power previously only associated with larger systems, the SGI MaxImpact which is the basis of our design.

2.2 State-of-the-Art

This section covers the best CAVE setups that are currently being used. Many of these are direct descendants of the original EVL CAVE and the subsequent company which was formed from their research.

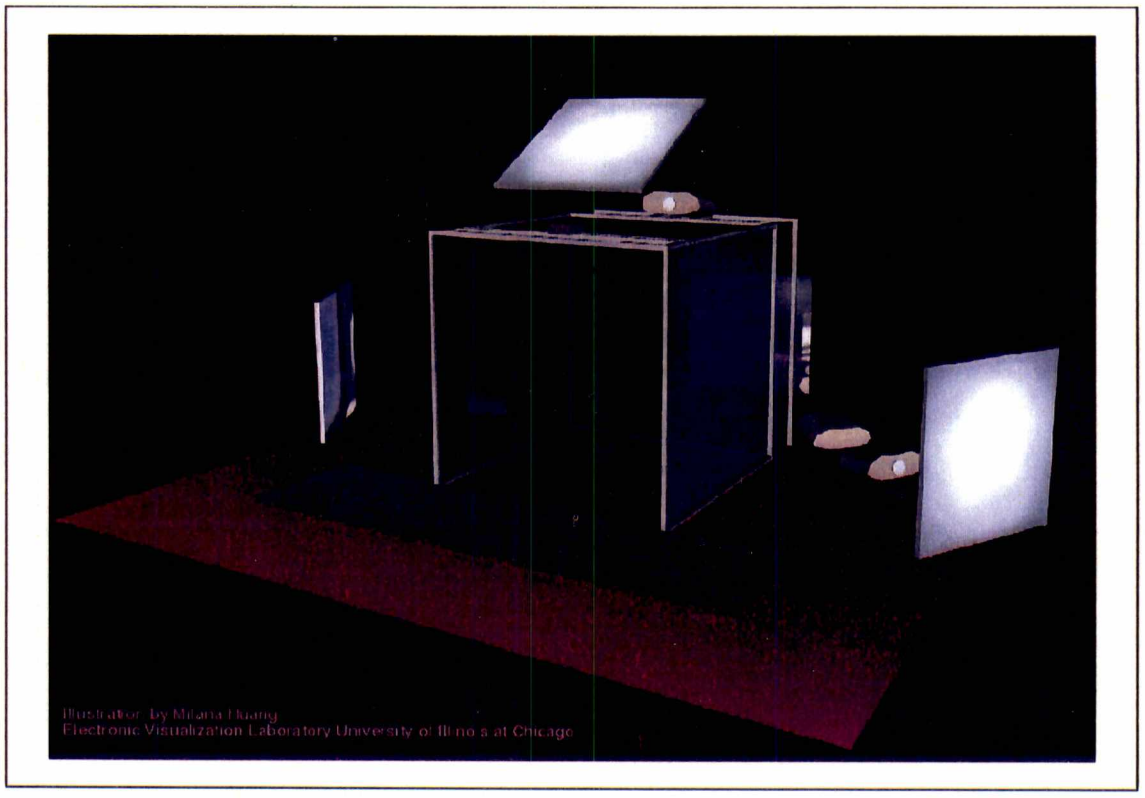


Figure 2.2: *Four projector CAVE in place at EVL (image courtesy of EVL, University of Illinois, Chicago).*

2.2.1 Electronic Visualization Laboratory (NCSA)

The concept of large surround screen interactive displays was first shown at EVL. Current research now involves creating a high-speed network for use in CAVE-to-CAVE communications [54], known as the I-WAY. This will allow not only multiple people to visualize a project at the same time in one CAVE, but multiple people in multiple locations to collaborate on research. A model of the current setup at EVL is shown in Figure 2.2. This system is a four projector system, projecting onto three walls and the floor.

2.2.2 Iowa Center for Emerging Manufacturing Technology

The CAVE at the Iowa Center for Emerging Manufacturing Technology, built by Dr. Cruz-Neira formerly of EVL, is a direct descendant of the first CAVE located at EVL. Iowa is studying the level of immersion needed to visualize a particular project, including architecture, chemistry,

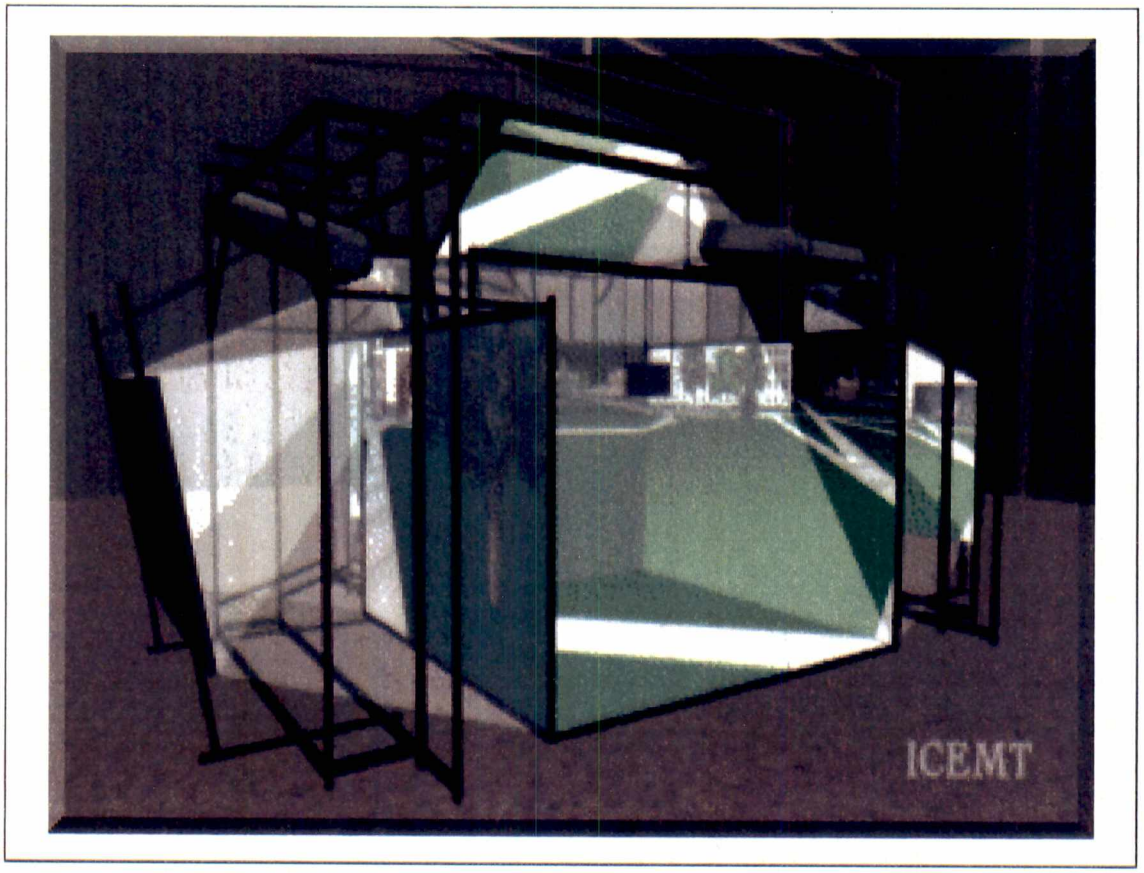


Figure 2.3: *Four projector CAVE in place at Iowa State University (image courtesy of Iowa State University).*

physics, and statistics. This CAVE setup is shown in Figure *2.3* and is also a four projectors system, projecting onto three walls and the floor.

2.2.3 Stanford Computer Science

The Responsive Workbench is a collaborative effort between Stanford's Computer Graphics Lab and the German National Research Center for Computer Science. This device is similar to the CAVE in that it is a large screen display which can be viewed by several users at the same time. However, the display is on a table measuring six feet by four-and-one-half feet (see Figure *2.4*). The display is a mirrored rear-projection image displayed from an SGI Onyx InfiniteReality workstation. Fakespace, maker of the BOOM HMD, is marketing the workbench as the "Immersive Workbench" for medical training, product design, and other simulation research. Recently, a two

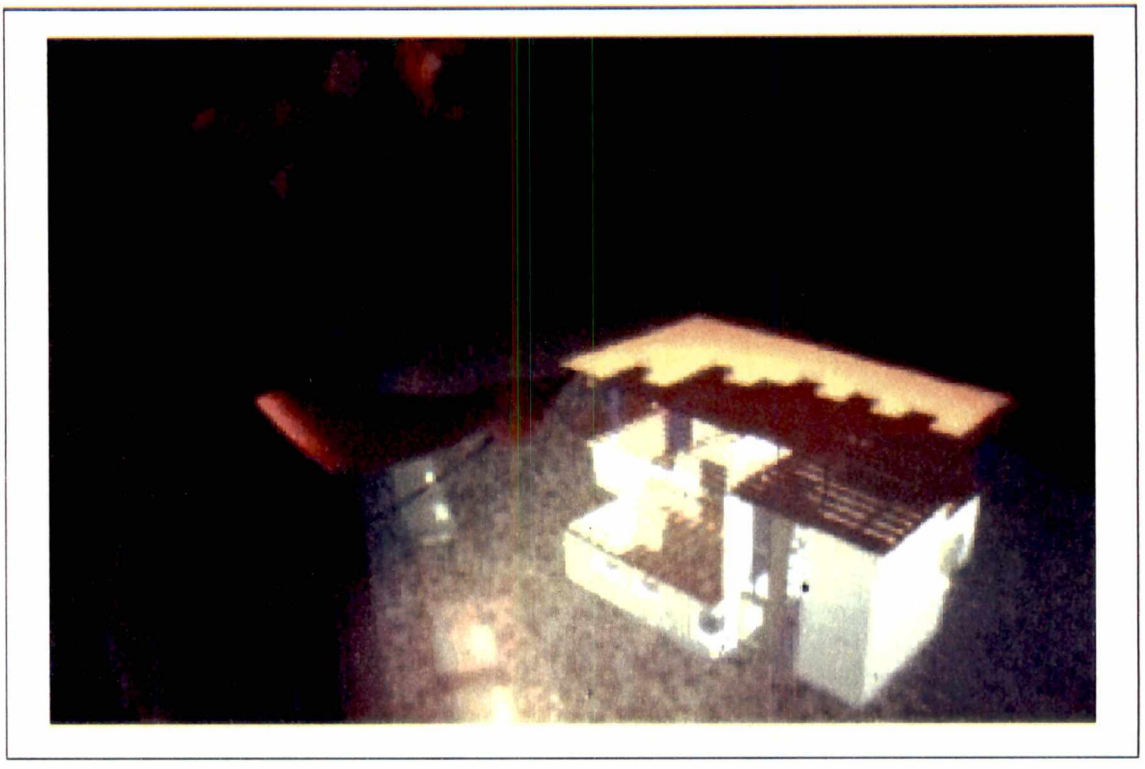


Figure 2.4: *The Responsive Workbench, a large screen table display (image courtesy of Stanford University).*

user version of the workbench was presented. This is accomplished by displaying two pairs of stereo image, one for each user [3].

2.2.4 Massachusetts Institute of Technology

Most VR systems in use make use of HMD's and data gloves. These devices can be cumbersome. To alleviate this problem Russel [87] suggests using passive devices, such as cameras, in an interactive virtual environment (IVE). Here a large projection screen is used in conjunction with passive sensing. A camera and microphone are used to input gestures from the user. For this system, three SGI workstations are used: one for sound input, one for video input, and one for video display. The gesture driven system is slower than when using more conventional input devices, but the user does not have to "suit up" with special hardware and is not "tied down" with wired devices.

2.2.5 Other Large Screen Deployments

Immersion can take on many different forms, as shown by the Iowa driving simulator [61]. Here a high-fidelity, fully immersive, interactive environment has been built complete with visual, audio, tactile, and force feedback. A screen surrounds a mock-up of a car to provide immersion for the user.

Caterpillar was one of the first manufacturers to use VR in the design of its products. The CAVE which they have installed is built around a full size mock-up of a back-hoe. This allows the designers to test new concepts in the virtual world before actually spending money to build a complete test model.

Several other large screen projection virtual reality displays similar to the CAVE have also been developed with names such as the CyberStage, Vision Dome, Visionarium, and Mirage. All of these are derived from the same basic idea of projecting a computer generated image onto a large screen so multiple users will be able to be immersed in the virtual environment. The system we have built is no different in this aspect. The next several sections describe the hardware used in our system and the uniqueness of the setup starting with the hardware used for user interaction and ending with the large screen display.

2.3 VR Hardware Interface for MERLIN

For 3D data interaction several pieces of VR based equipment are used. This includes a Virtual Technologies' *Cyber*Glove, a Polhemus Fastrak, and a Spacetec Spaceball. Each of these are serial devices and are connected as shown in Figure *2.5*.

The *Cyber*Glove from Virtual Technologies [111, 57] (see Figure *2.6*) is a high-end data glove. This includes a glove equipped with 22 bend sensors to measure the motion of the hand and fingers along with the *Cyber*Glove Interface Unit (CGIU) to provide a serial interface to the glove. The sensors used in the glove allow it to easily track the bending of a joint. The output voltage of each sensor varies linearly with the change in bend angle so there is no resolution loss near the limits of a joint. The CGIU provides amplification and digitization circuitry to give 8-bit resolution output for each sensor. The range of angle values can be set through glove calibration. An offset and gain for each sensor is set in the CGIU. The CGIU has a single-pole analog low-pass filter with a corner frequency of 30 Hz in series with each sensor. Each of the fingers has 3 bend sensors to measure the metacarpophalangeal (MCP), proximal interphalangeal (PIP), and distal interphalangeal (DIP) joints. These are the joints where the finger joins the palm, the second joint from the finger tip, and the joint closest to the finger tip, respectively. The thumb has 2 bend sensors for the MCP and the IP joints. Abduction sensors to measure the amount that the fingers move laterally are provided for the thumb, middle-index, ring-middle, and pinkie-ring fingers. Additional sensors measure the thumb's rotation across the palm, the pinkie's rotation across the palm, and the pitch and yaw of the wrist.

The Polhemus Fastrak® [73, 14] (shown if Figure *2.6*) is an electromagnetic, six degree-of-freedom (DOF) tracking instrument. This consists of a System Electronics Unit (SEU) which allows for serial communication with a transmitter and up to four receivers. The tracking system employed by the Polhemus Fastrak uses electromagnetic fields to determine the position and orientation of an object. The transmitter generates near field, low frequency, magnetic field vectors from an assembly of three colocated, stationary antennas. The receivers containing a single assembly of three colocated, remote sensing antennas. The signals sent from the transmitter are

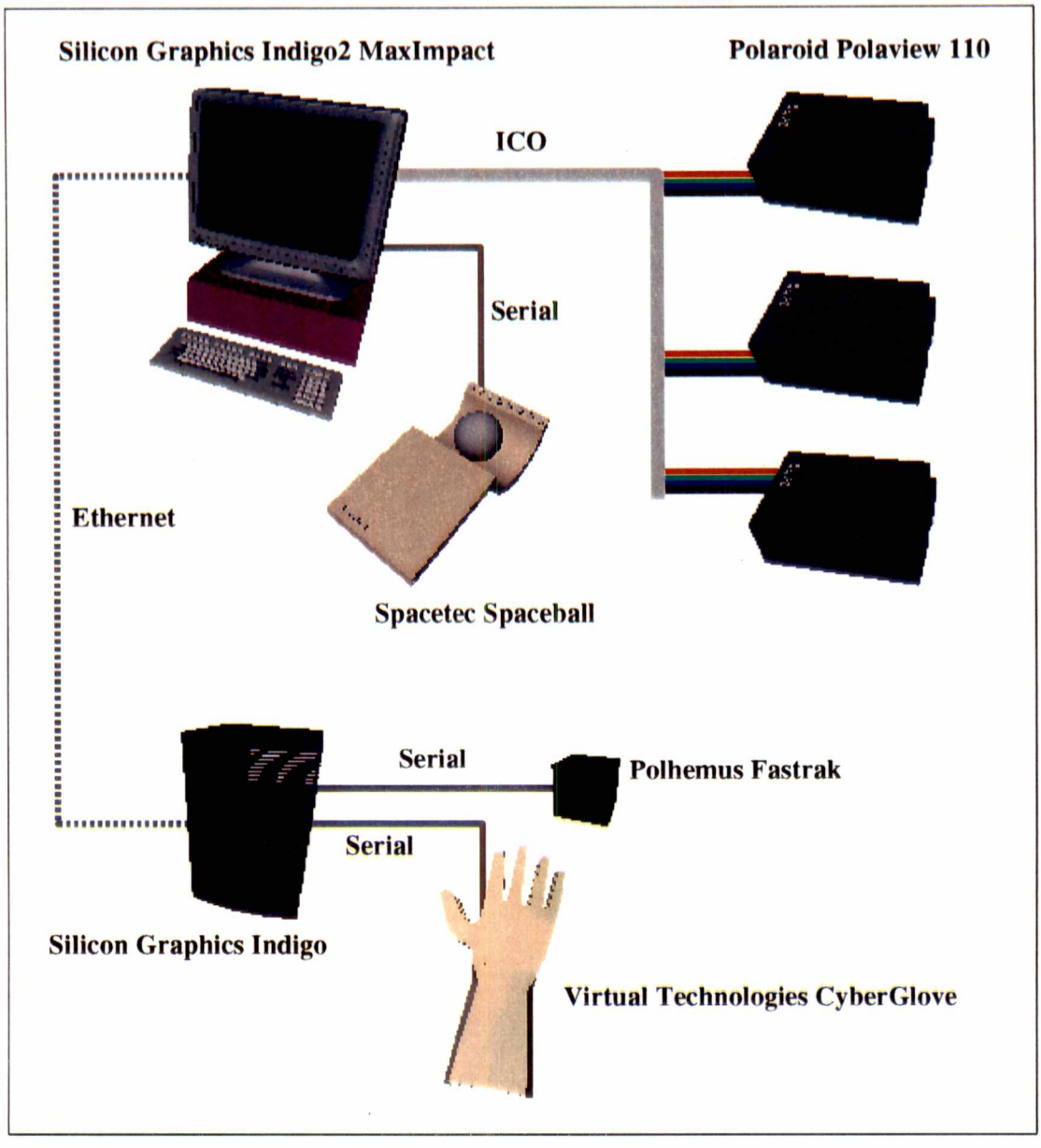


Figure 2.5: *Connections of all hardware used for user interaction with the data.*

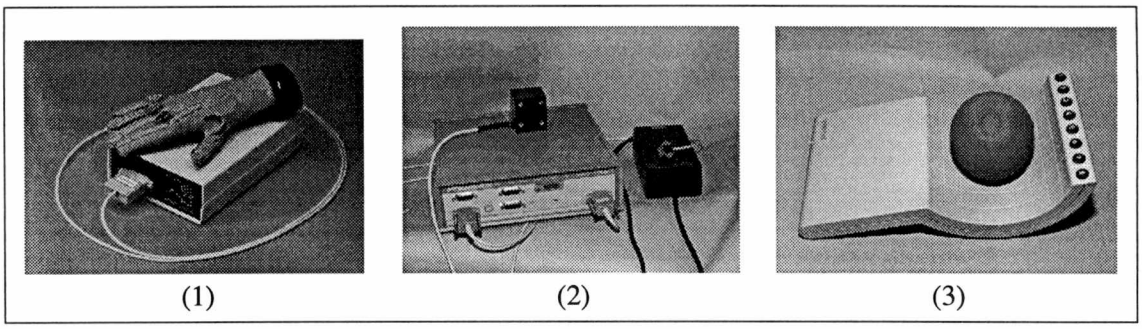


Figure 2.6: *VR hardware used including (1) a Virtual Technologies'* Cyber*Glove, (2) a Polhemus Fastrak, and (3) a Spacetec Spaceball.*

received and input into mathematical algorithms to compute the relative position and orientation of the receiver with respect to the transmitter. The Fastrak claims 0.03" RMS static accuracy for position and 0.15° RMS for orientation of the receiver while providing 0.0002 inches/inch and 0.025° resolution within the operating range of the transmitter, 30 feet. There is a 4.0 millisecond latency from the center of the receiver measurement period to the beginning of the output transfer.

The Spaceball is a popular ground-based input device (see Figure *2.6*). This device uses strain gauges to measure six DOF. It has 0.1" positional accuracy and 0.5° orientation accuracy. The sensitivity for each of the degrees-of-freedom can be set by the user. The device also gives the user 9 control buttons.

The interfacing of the VR hardware used for interaction presents a challenge in that all three devices are serially controlled and, unfortunately, the SGI machines used to interface them only have two serial ports. For this reason another SGI machine is used solely for its two serial ports while the user interface resides on a machine using one of its serial ports. Open Inventor has an interface for the spaceball available through a built-in node, so the spaceball is connected to the SGI MaxImpact that drives the projectors and contains the GUI. In this manner, the Spaceball is accessed via the Open Inventor Class SoXtSpaceball with the Spaceball device driver installed on the first SGI (see Figure *2.7*).

The DataGlove and the Polhemus tracker are both connected by serial lines to the second SGI computer, an Indigo R4000. Using the second computer also helps to offload some functions from

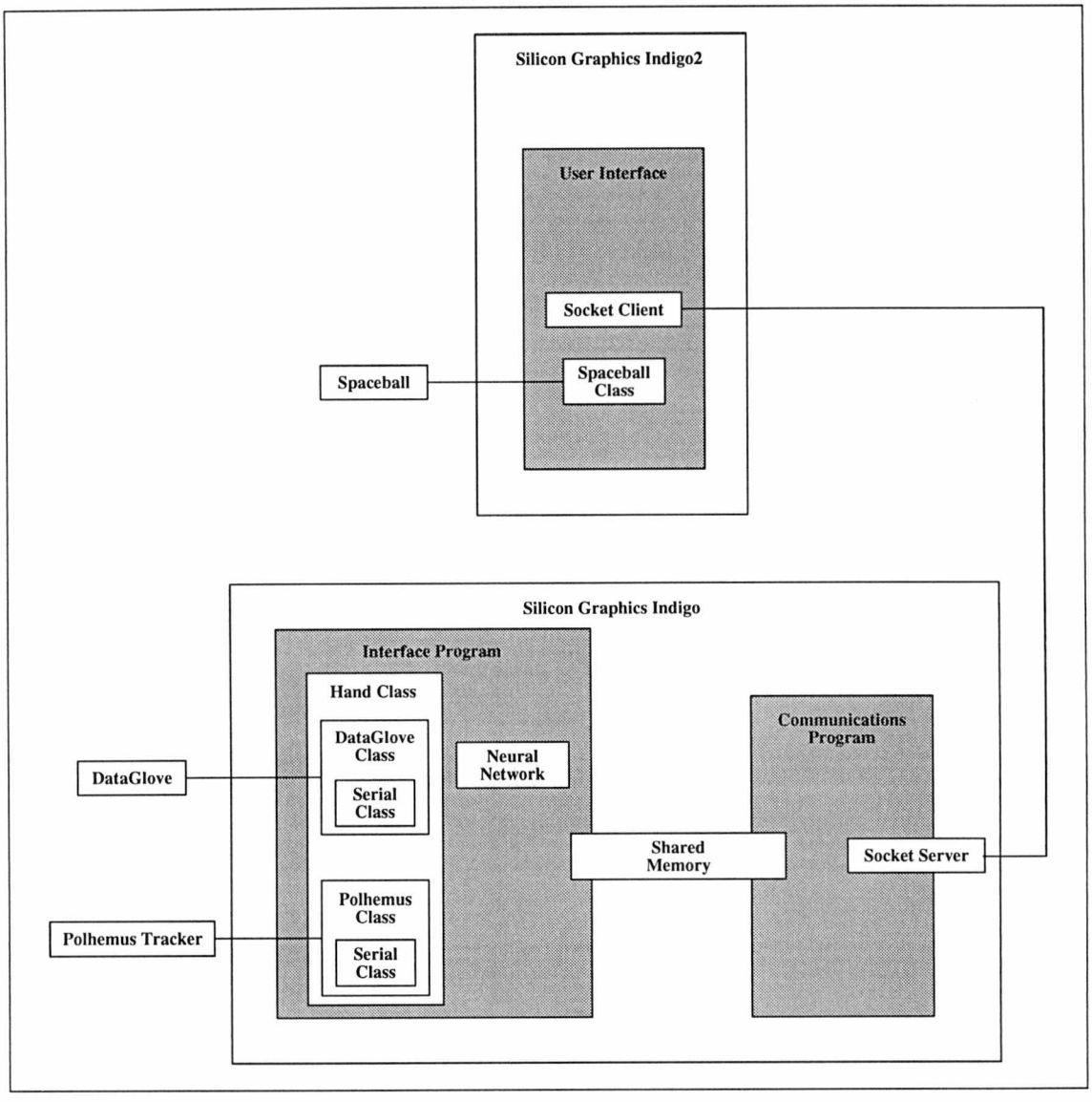


Figure 2.7: *Diagram showing the acquisition of data from the DataGlove and Polhemus tracker and the communications used to send the data to the user interface.*

the MaxImpact. All communication with the Polhemus Fastrak and *Cyber*Glove, however, must be custom written, including the low-level serial communication with the VR hardware since the software that is shipped with the devices are PC based. The hardware interaction software for this project is developed in an object-oriented fashion using C++ class structures so it can be easily ported to other applications if needed. To interface these devices, first a *Serial* class was written to handle all of the low level serial setup and communication between the SGI and the serial ports. Each device is read in binary mode at 38,400 baud, the maximum speed the SGI ports are capable of handling. Also, since these devices are built with a standard PC serial interface, special serial adaptors had to be built due to the different pin-out of the SGI serial ports. A second layer of classes was then written for each device in the form of a *Polhemus* and *Glove* class. These are derived from the Serial communications class which contains the basic serial I/O functions. They are then wrapped into a *Hand* class so that one instance of the class can control both devices with a high level interface. The reading of the sensors and transfer of the data to the user is then controlled by two separate programs which communicate to each other via shared memory. The first is an interface program directly to the hardware which reads the serial devices in a tight loop at the highest rate possible and communicate that data to the host program via the shared memory. Two programs allow the host program to achieve the highest communication rate possible and not be bottle-necked by the speed of the serial interface. The twenty-three bend sensors on the glove and the six degrees-of-freedom from the tracker are returned to the hardware interface program. The data from the glove is then passed through a neural net to recognize the current posture. The neural network used for determining the posture is currently implemented using *Matlab's* neural network toolkit. The network is a fully connected feed-forward network trained using back propagation. There are 23 input nodes to the network. These are connect to 15 hidden nodes which subsequently connect to five output nodes. The values of the output nodes are thresholded at 0.5 to create a five digit binary code capable of up to 32 different postures. The posture along with the orientation and position information is then written to the shared memory. The host communications program handles passing the information from shared memory to the other machine. It is a socket server which passes the most recent posture and position along to any client on the system.

2.4 Design of Our CAVE Setup

The CAVE is considered to be the state-of-the-art in visualization technology. It relies on the best computing, audio-visual, and virtual-reality hardware available. Therefore, when building a CAVE the first thing that comes to mind is price. Building a CAVE is a very expensive project to undertake. One of our goals was to design and build a low-cost usable CAVE. Building on existing CAVE's, we took the features that we needed for visualization while staying within the limits of our budget and also leaving enough room for future expansion. The CAVE's display consist of three parts: the screens, the projectors, and the graphics engine. Several trade-offs had to be made for price versus performance. For example, most other large screen display setups employ some type of 3D spatially located audio to help enhance the visualization experience. It was decided not to have any spatial audio for this setup. Other trade-offs are described in the following sub-sections.

2.4.1 The Screen

The screen in the CAVE is used to immerse the user and can completely surround him. Some are setup like a small room with images displayed on all the walls, floor, and ceiling. Screens used can be flat, cylindrical, spherical, or parabolic. One other consideration is whether to use front or rear projection screens. Rear projection requires more space since the projectors and mirrors must be behind the screen. Also, when using a rear projection screen, the gains are lower so the projectors must be brighter in order to shine through the screen. However, using a rear projection system means that the user can not stand between the projector and the screen and block the view. These factors add to the cost of the screen, so we chose a front projection system. The screen used here is the only component that is not available off-the-shelf due to the size of the screen and the specification to have it adjustable for future expandability. It is the only custom piece of hardware in this setup designed and built by Stewart Filmscreen to meet our specifications. The screen was designed with a reconfigurable geometry in mind so that it could be used at variable radii and with varying geometrical setups. It also had to fit in a room that is twenty-two feet wide with nine feet

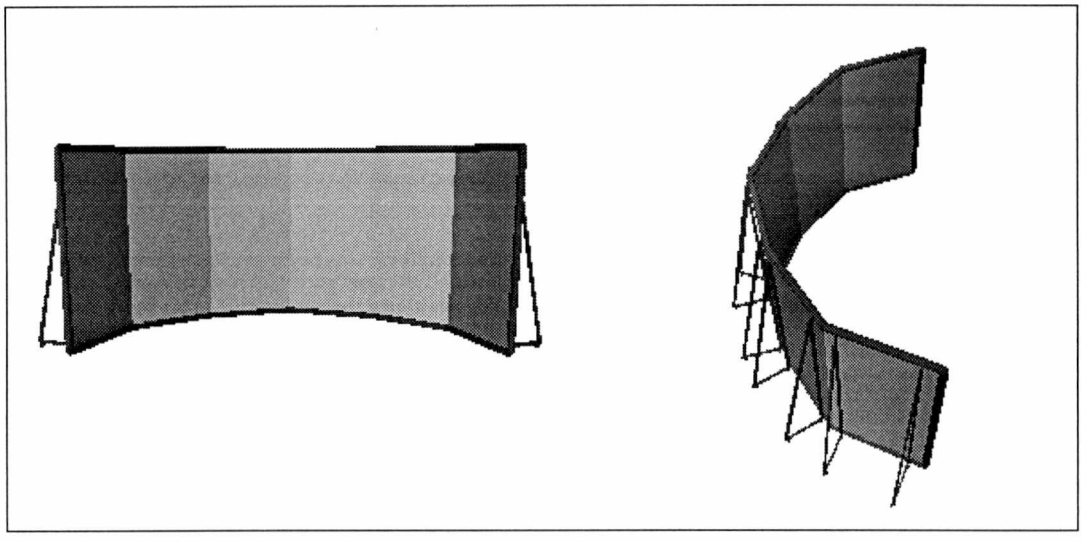


Figure 2.8: *Model of the reconfigurable screen shown with a radius for 120° circular view.*

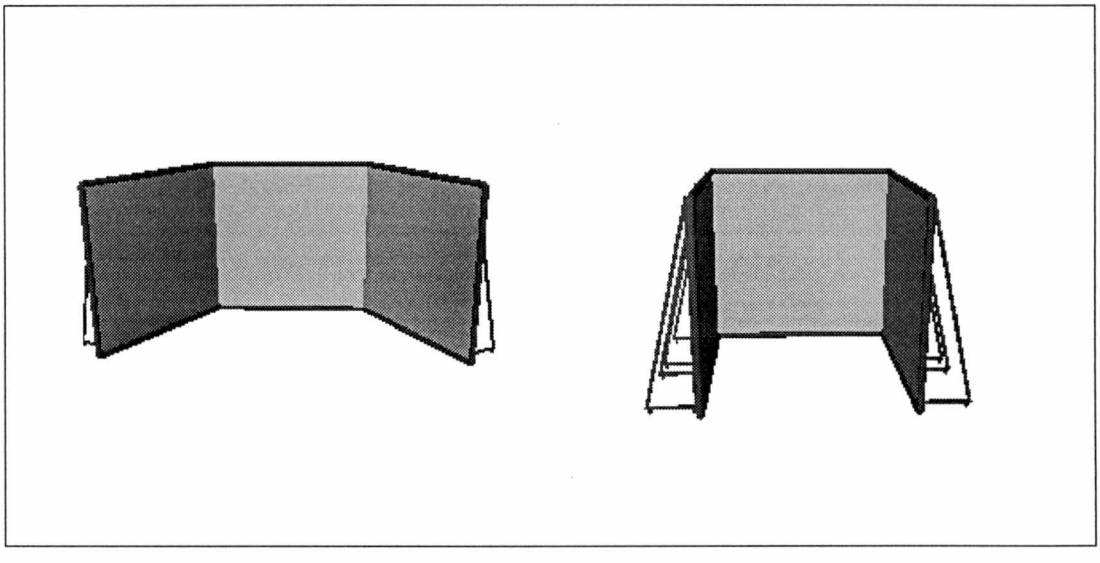


Figure 2.9: *Two other possible screen configurations.*

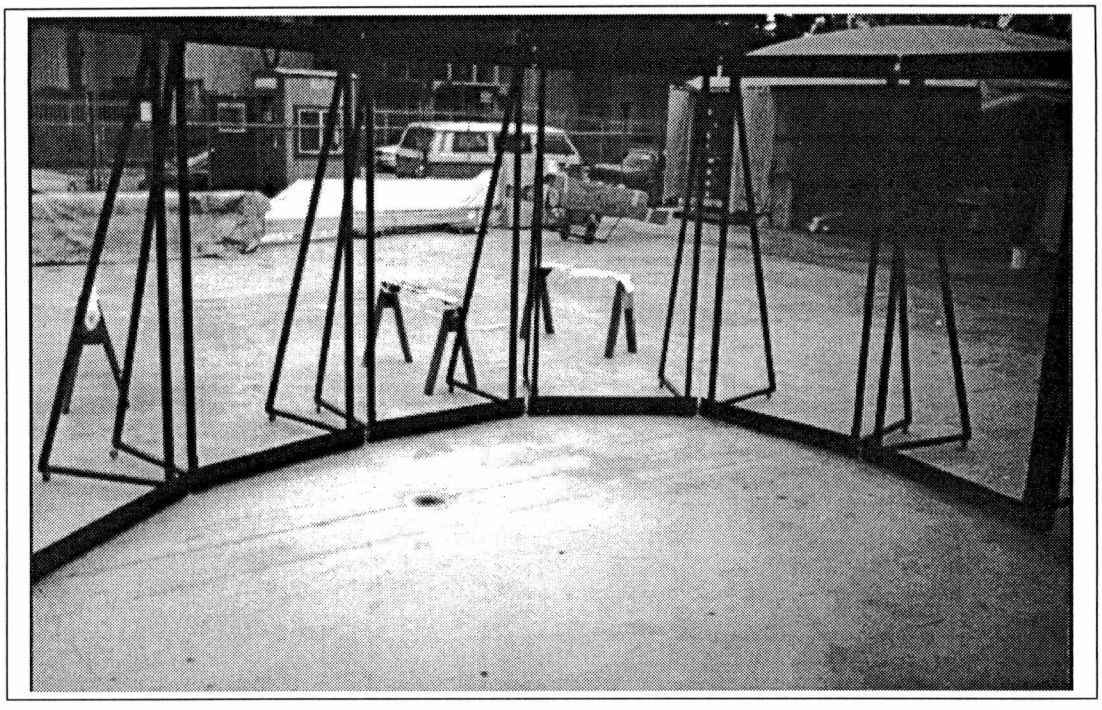


Figure 2.10: *The six frames hinged together to form the large reconfigurable screen.*

ceilings. A model of the screen is shown in Figure *2.8.* with two other possible configurations shown in Figure *2.9.* The screen itself consists of one seamless piece of material eight feet and two inches tall by twenty-seven feet and four inches wide stretched across six frames each eight feet and ten inches tall by four feet and six inches wide. Each frame (see Figure *2.10*) is hinged to the next making several setup configurations possible. The overall frame size is eight feet and eleven inches tall by twenty-eight feet wide. To keep the screen from bowing out between the hinged frames, a small piece of $\frac{1}{16}$" tension cable is run to pull the screen tight at each joint. The cables are painted to match the screen color so the seams are not as visible. Each frame is back braced and on one inch casters for mobility. The top, bottom, and sides of the screen have a six inch masking. The screen material is an ultramatte fabric with a gain of 1.5 and gives an overall viewable image size of twenty-seven feet wide by seven feet and ten inches tall. The screen had the least trade-offs for price versus performance. Due to space considerations as well, a front projection system is used. But the screen is reconfigurable for future use.

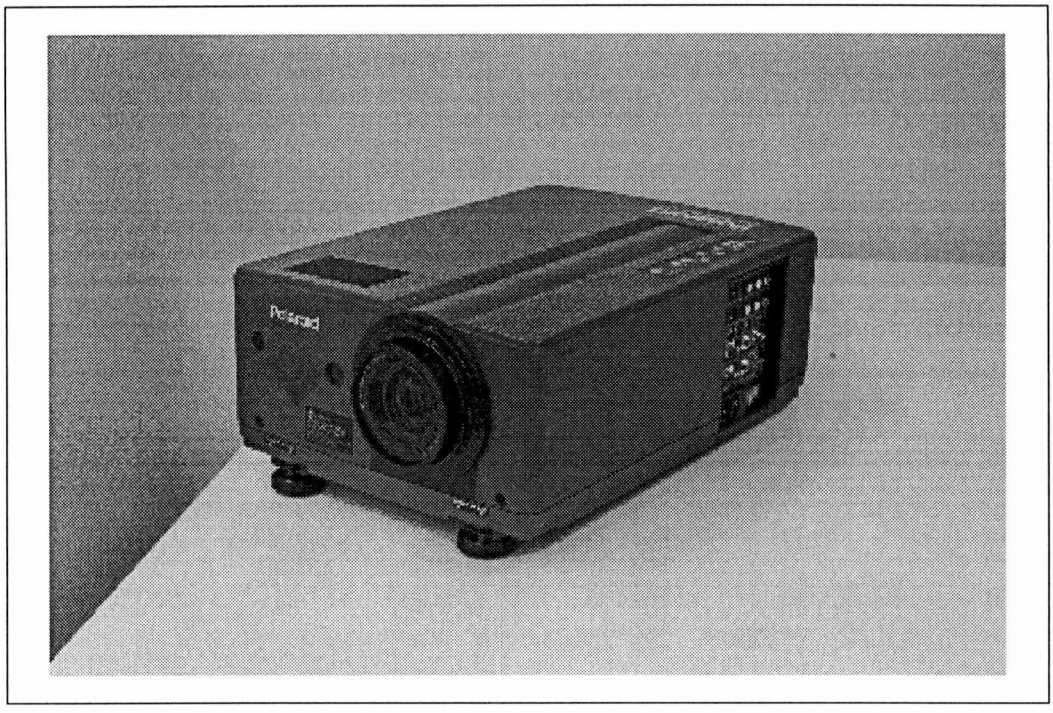


Figure 2.11: *Polaroid Polaview 110 LCD projector.*

2.4.2 The Projectors

The projectors that are used in most CAVE systems are usually modified to correct optical distortions or to aid in edge blending. Also, some setups contain special edge blending hardware to create a seamless image since the key to the projection system is the seamless blending of the edges of the images. Optical modifications and edge blending hardware add greatly to the cost of a projection system. For our system, the key features at which we looked included:

- Resolution,
- Brightness,
- Number of colors, and
- Update refresh rate.

For projection, we are using three Polaroid Polaview 110 LCD projectors. These are standard off-the-shelf multimedia projectors. They contain three polysilicon liquid crystal panels and a

250 Watt metal-halide lamp each. They are capable of 640x480 resolution at 24-bit color. The brightness is rated at 500 ANSI lumens. These are driven from the SGI ICO at 60 Hz with a VGA sync. These units are used unmodified and offset from the screen fifteen to eighteen feet. Most high-end systems have modified projectors to perform the blending or special hardware is also available to achieve the same results. Many projectors also have modified lenses to correct for optical distortions created from projecting onto screens which are not flat. For this setup, edge blending is performed in software. The projectors must be set-up to interlace or merge three images into one continuous image. More about the projector setup is given in Section *2.4.4*. Each projector has settings for the horizontal and vertical size of the projected image, along with a separate zoom. To avoid vertical scan misalignment the horizontal size must be set to 800. For final alignment, software adjustments must be made. We are not currently worried about perfect edge blending. What is done will be implement via software. Any optical distortion corrections will also be made using software. These projectors are not capable of the update rates needed to produce high frame rate stereo images, but they do handle the maximum resolution and frame rates the graphics engine is able to produce when driving more than two projectors. Each projector is mounted on a tilt mechanism which also has a height adjustment. These are then mounted on sturdy carts which gives the ability to quickly adjust the position of the projectors for proper edge matching.

2.4.3 The Graphics Engine

The graphics engine for the CAVE's display is what drives the signal to the projectors. All CAVE's that have been built to date are powered by Silicon Graphics Onyx computers using either Reality Engine or Infinite Reality graphics hardware. We also use a Silicon Graphics to drive the projectors for the display since it has the highest performance graphics engine available. However, our CAVE is unique in that its graphics are powered by a desktop computer instead of a larger rack machine. The projectors are currently driven by a Silicon Graphics Indigo2 Maximum Impact with the Impact Channel Option. This machine is the first desktop package with built-in hardware texture mapping and provides the highest-performance textured, 3D graphics available in a desktop box.

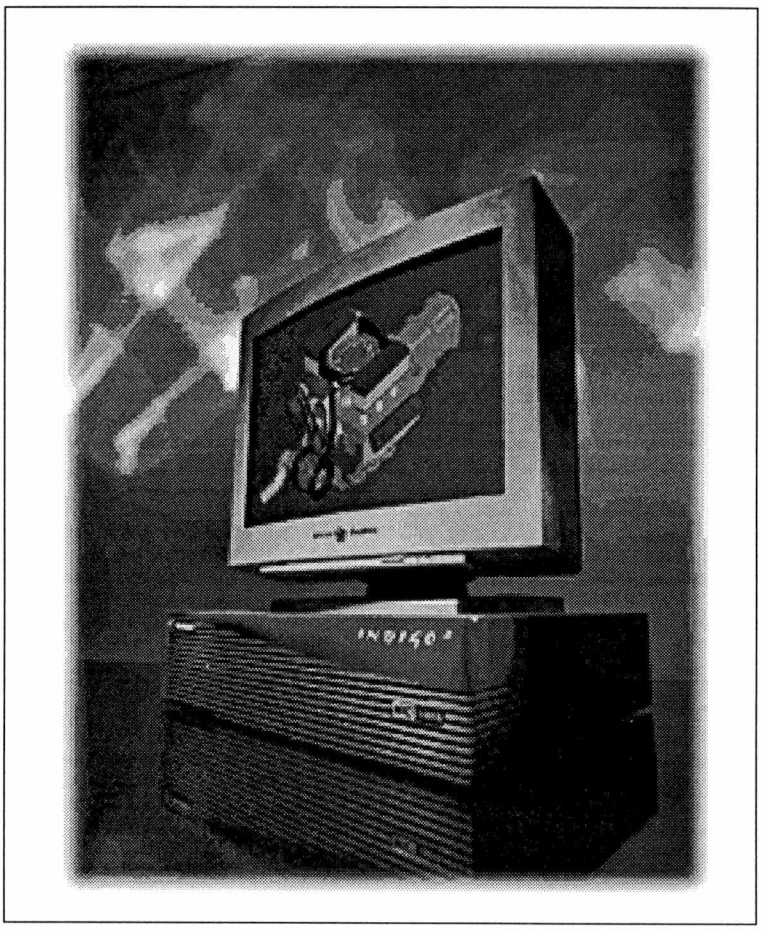


Figure 2.12: *Silicon Graphics Indigo2 Maximum Impact with Impact Channel Option.*

This machine has:

- 195MHz MIPS R10,000 processor
- 128MB RAM
- Maximum Impact graphics with 4MB texture memory
- Impact Channel Option

The ICO gives the capability of four 640x480 channels for use in displaying the Virtual Environment. Three are used for the CAVE's display and one 640x480 display for the user control interface. There are some advantages to using a desktop computer. The one obvious advantage is cost. While most existing CAVE's cost millions to build, ours is only a fraction of that cost. We do sacrifice some of the speed, but can easily scale up later as computing prices fall and performance increases.

2.4.4 The Desktop CAVE

We have designed a low cost Desktop CAVE. It is built mostly with off-the-shelf, unmodified hardware [38]. Again, the setup is known as the *m*ulti-us*e*r *l*ow-cost *in*tegrated (MERLIN) visualization system. It is comprised of the custom built screen, three projectors, and an SGI to drive the video for the system. Before building the system, it was designed using a virtual model (see Figure *2.13*). This helped achieve the optimal size and design of the screen before it was built. Also, the position of the projectors could be tested to determine if they would be in the line of sight of the users since a front screen projection configuration is used. With the hardware design and layout confirmed by the virtual model, the physical setup of the system could proceed.

Our initial setup has the screens configured as three flat screens with a 160° field-of-view (see Figure *2.14*). This is just one example of the many radii that are possible with the six frames, from a completely flat screen to a closed hexagon. Setting up the projectors to give a seamless image presents the biggest challenge of our design since we have no custom hardware for edge-blending. The position and orientation of each projector must be fixed along with the focus and zoom. Using the same coordinate system as before, the x position must be centered with the screen.



Figure 2.13: *Layout of the room containing the desktop CAVE consisting of three projectors driven by an SGI MaxImpact and projecting onto a custom-built screen.*

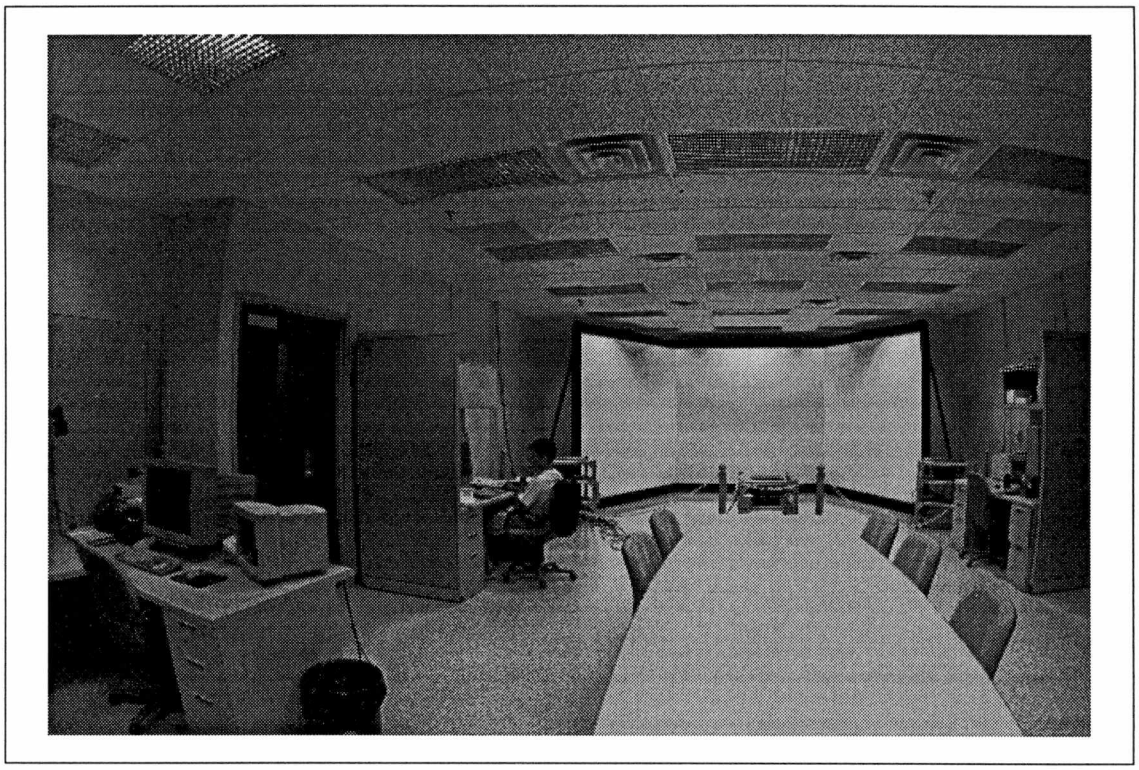


Figure 2.14: *Fish-eye view of the room housing the CAVE showing the current setup.*

The rotation about the x axis is determined by the keystone correction built into the projectors. Keystone correction allows a projector to be tilted and still project a rectangular image. Here, it is fixed at 8.8° so the projectors must be rotated by this amount about the x axis to provide a square image. The keystone correction, along with the height of the screen, determine the position along the y axis since it must be tilted 8.8° and centered on the screen. To center the image on the screen, the projector must be placed 2.3 feet from the center of the screen. Also, the rotation about the y axis must be 0° to give a square image. The position along the z axis is not pre-determined. For the size image needed, taking into consideration the zoom characteristics of the projectors, it must fall between 189 inches and 304 inches from the screen. The orientation must be such that the projector is perpendicular to the screen onto which it is projecting. After the projectors are positioned, then the focus must be adjusted individually for each projector to give the sharpest image.

The projectors also have the capabilities to adjust the horizontal and vertical position of the image on the LCD itself. To allow the entire image to be viewed the horizontal position must be set to 90 and the vertical to 97. Also, so that the image does not appear jittery or fuzzy the horizontal phase must be set to 0 and the horizontal size to 800. Unfortunately, these projectors do not have any adjustments for color, so some color differences across the screen can be seen.

2.4.5 Future CAVE Hardware Consideration

This was the proof-of-concept design. The goal was to design and build a low cost visualization system. This system, however, could be improved dramatically with the use of better hardware. One limitation is the lack of stereo viewing capabilities. This is due in part to the projectors, but mainly to the capabilities of the graphics engine. Stereo viewing would allow the user to see the models in true 3D since each eye is shown a different perspective of the model. This increases the sense of immersion felt by the user. For true stereo viewing (using shutter glasses), the machine driving the projectors needs to be capable of at least 96Hz refresh rates, 120Hz would be preferable. The projectors also have to be capable of handling these rates. Also, higher resolution displays would show more detail. This again requires a better graphics engine and projectors to support the increased resolution. Another improvement would be to use brighter projectors so the models can be viewed with the lights on. Also, using backprojection screens would allow the user to get closer to the screens without blocking the image being projected. This requires new screens which are backprojection capable and a much larger area to allow room behind the screens for the projectors and their folded optics. Each of these upgrades, however, comes with a significant price tag.

2.5 User Interface Design for Large Screen Display of Models

Several constraints are put on the design of the graphical user interface for this project due to the hardware used. The ICO is configured to display four 640×480 screens, one in each quadrant. Three of these are tiled together to form one large image to be projected and the GUI resides in the remaining quadrant. The ICO uses a 1280×960 frame buffer [69] (see Figure *2.15*). Each

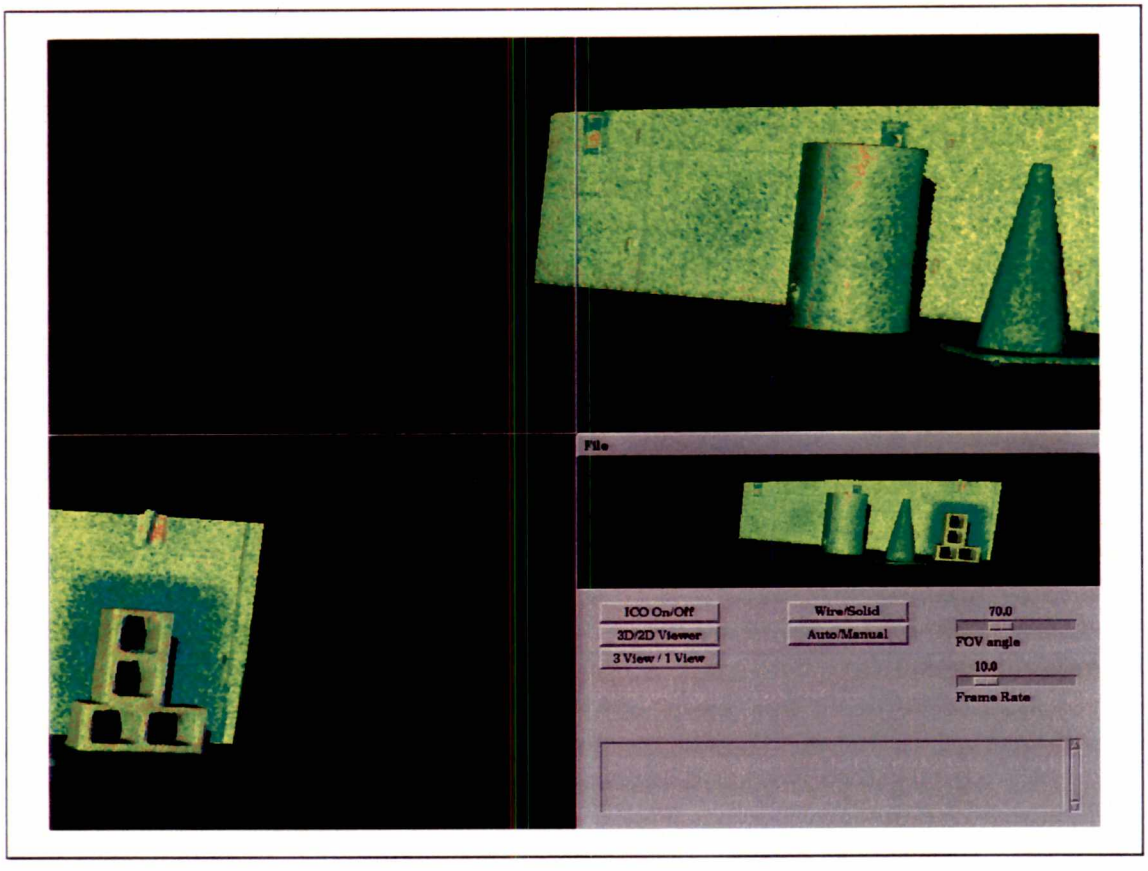


Figure 2.15: *View of the ICO frame buffer showing three views of same scene to create one seamless view to be displayed with the projectors along with one view for the user interface. The model shown was created from several sets of range data supplied by Oak Ridge National Laboratories acquired by a Coleman laser range scanner. The model is texture mapped with color coded quality values returned from the scanner.*

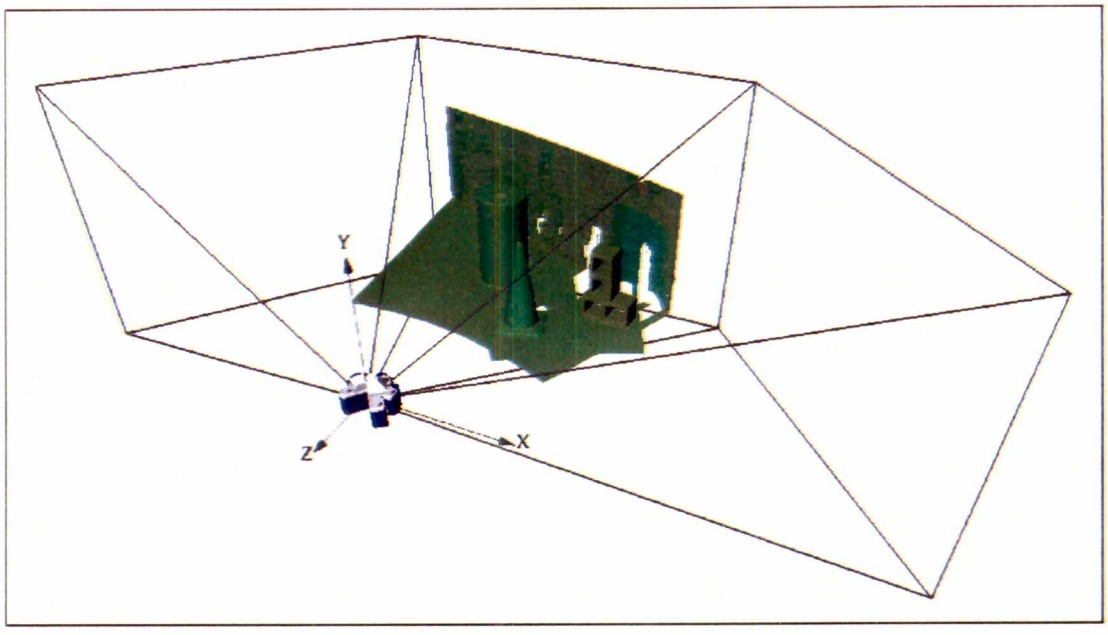


Figure 2.16: *Three cameras in one of many possible configurations all looking at the same scene to create one continuous view.*

640×480 quadrant of the buffer is output to a separate display. For visualization, the three buffers sent to the projectors create a 1920×480 display on the screen. To accomplish a continuous display, camera models must be setup properly to create a continuous image of a single model across the three displays since we use edge matching instead of edge blending. Three identical perspective camera models are placed at the same point. One camera is rotated about the y axis by the field-of-view angle, another by the same amount in the opposite direction. This creates three non-intersecting views from the cameras, forming one large perspective camera as shown in Figure *2.16*. From a hardware point-of-view the projectors must be aligned such that the edge pixels are adjacent and the horizontal rows align, as described in Section *2.4.4*. These Open Inventor camera models, along with the projector alignment, take care of the image merging to form a complete and seamless image display on the large screen. The final quadrant is output to a screen at the user's workstation. From here the system can be controlled via a user interface. In addition to the seamless image, since four separate Open Inventor viewers are used, each screen can display a separate view-point of the model. For example, top, side, and front projections can be viewed simultaneously.

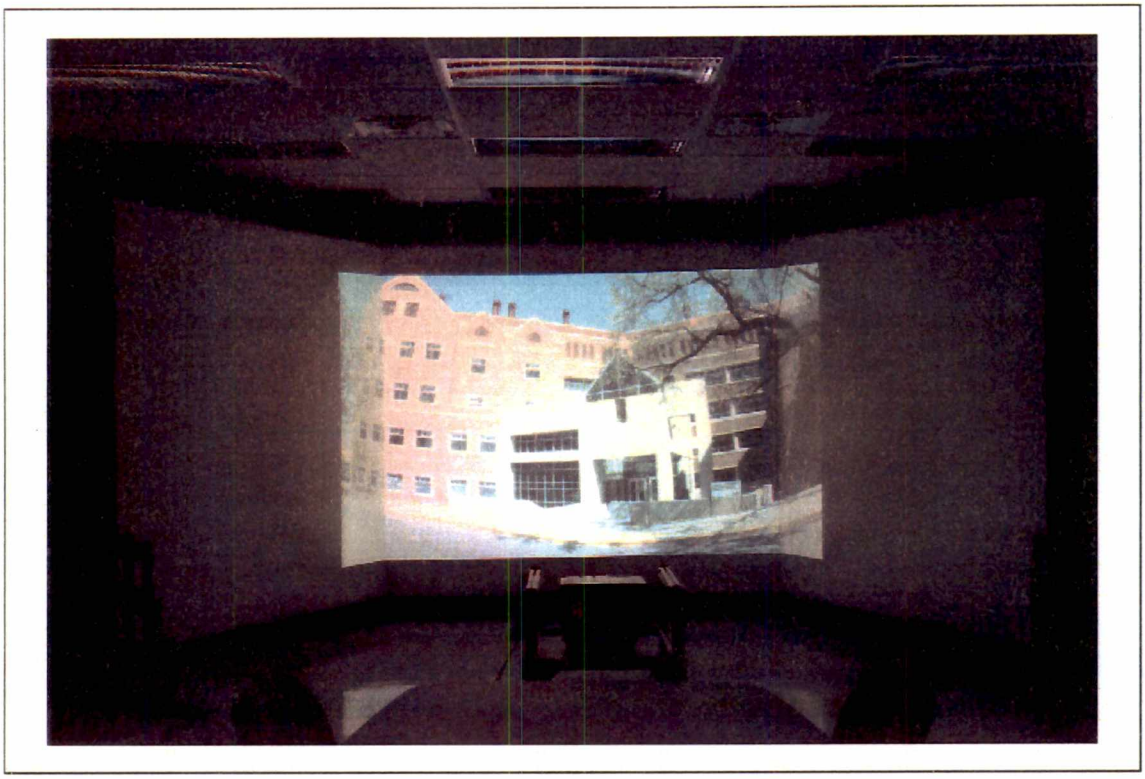


Figure 2.17: *High resolution image displayed on the CAVE.*

2.5.1 Viewing Large Images

Along with 3D models, we also need the capability to view very high resolution 2D images on the CAVE display, including 3000×2000 pixel images from high resolution digital cameras and 7000×4000 pixel images from image tiling. Figure *2.17* shows an example of one of these images. In a manner similar to the 3D data viewing, the screen must be split into quadrants, with the upper left quadrant containing the left of the image, the upper right containing the center, and the lower left containing the right of the image in order to display a continuous image across all three screens. The display interface was written using SGI's image library which allows a variety of image formats to be read. The same user interface used to display the 3D models is also used to display the images by switching to the 2D image mode using a button on the control panel (see Figure *2.18*). The user is also able to pan and tilt the image using the supplied buttons as well as zoom in and out on the image. A zoomed view of the previous of the high resolution image is

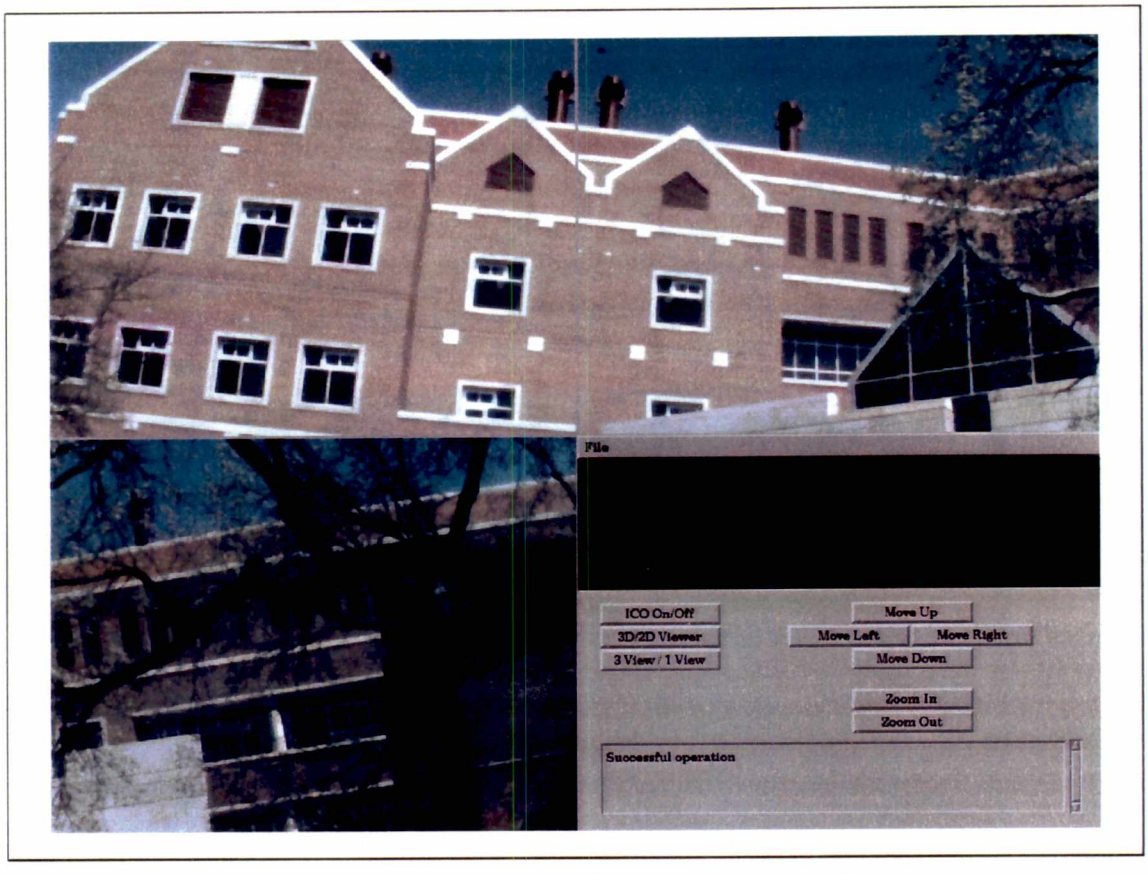


Figure 2.18: *View of the ICO frame buffer showing three views of same image to create one seamless view to be displayed with the projectors along with one view for the user interface.*



Figure 2.19: *High resolution image displayed on the CAVE.*

shown in Figure *2.19*.

2.5.2 360° Image Viewing

With the perceptron laser range scanner mounted to a pan/tilt mechanism, it is possible to obtain full 360° spherical images by tiling several images together. By using texture mapping, viewing the returned intensity image is very simple. A sphere is created, and the warped intensity image is texture mapped onto it. By moving inside the sphere, the user can look around the image in any direction. The only limitation to this technique is the amount of texture memory in the machine being used for display. Some of the images are very large, on the order of 80MB, so the hardware may subsample the image in order to display it. An example of this is shown in Figure *2.20*

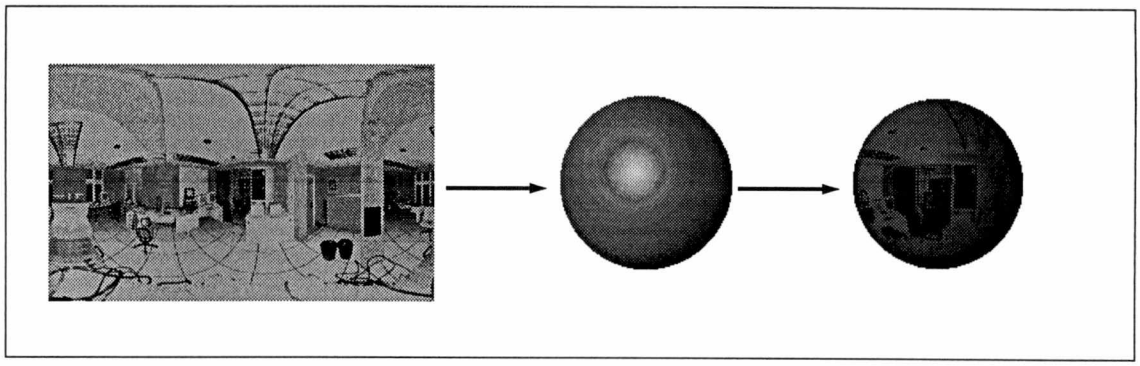


Figure 2.20: *Spherical image of the environment texture mapped onto a sphere.*

2.5.3 3D Model Viewing and Manipulation

The reason for initially building the CAVE was to view large 3D models. The ability to view large images and 360° images is an added bonus. Also, the data is manipulated using the VR equipment. The spaceball allows the user to "hold" the data and move it to any viewing position. Moving the data in this manner is more natural to the user than using a mouse and pressing a combination of its buttons to perform a translation or rotation. Figure *2.21* shows an example of a 3D model being displayed on the large screen. Since the software to display the models was written using the Open Inventor library, any Open Inventor model can be loaded and displayed by the system. Display of larger models has shown the limitation of this system, lack of speed. This is to be expected given the initial trade-off made in the design of the system of cost over speed and leads directly to the main research focus of the remainder of this dissertation – a means to quickly display large models while maintaining details.

2.6 Conclusions

To this point, we have described the design of a low cost, usable, large screen visualization display system for 3D data sets. The system has been built and is currently in use for display of models created from laser range data. Several configurations of the screen are available. Currently the system is set up as three flat screens giving a 160° field-of-view of the modeled scene. The

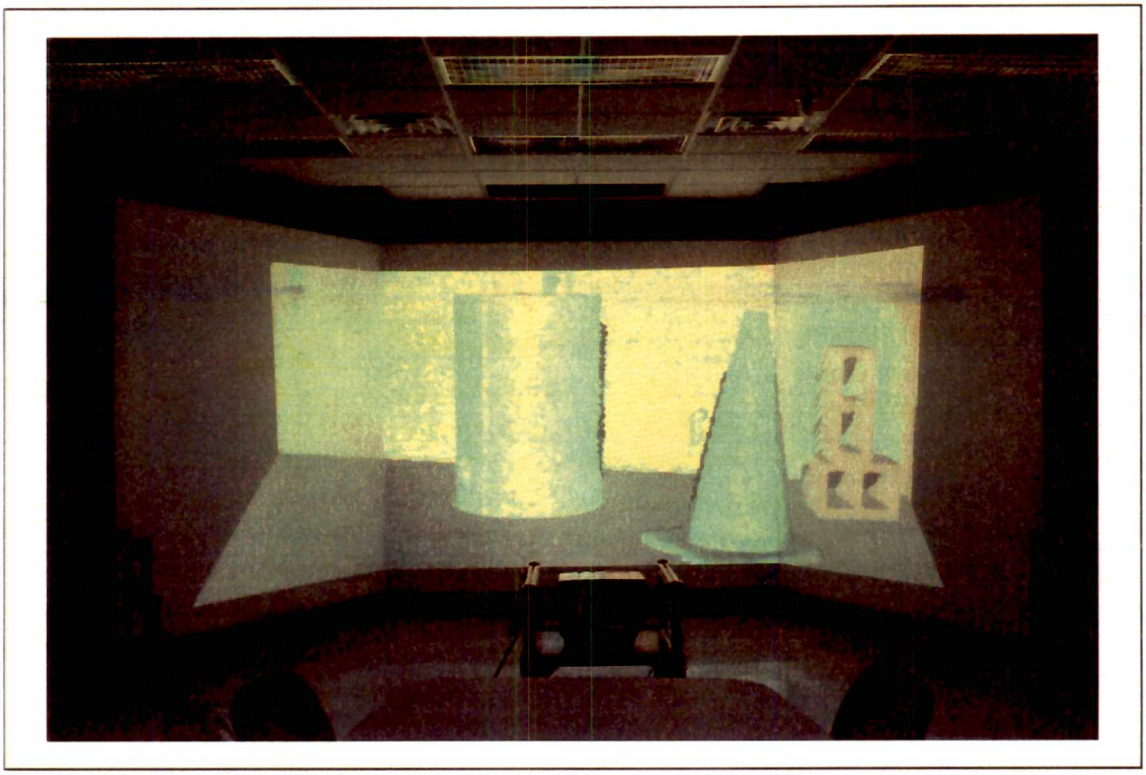


Figure 2.21: *Setup configured as 3 flat screens giving a 160° field-of-view displaying a model created from the laser range data supplied by ORNL.*



Figure 2.22: *Color difference visible at the seams of the screen where edge matching is performed.*

projector positioning is still being tweaked to give a perfectly seamless image. Close inspection of the screens shows one to two pixels of misalignment error visible near the bottom of the projected image at the seams. From the user's view-point, however, these misalignments are not perceptible. Also, a slight color difference is noticeable among the three projectors. As stated previously, the color is not adjustable on these particular projectors. Two of them are close in color matching, but the third is not. Figure *2.22* shows the seam where the color matching is off. The graphic performance is almost usable giving update rates of four hertz on texture mapped models with 16,000 triangles, two hertz with 30,000 triangles, 1.25 hertz with 63,000 triangles, and 0.7 hertz with 102,000 triangles. This translates to about 70,000 triangles per second. However, to achieve a frame-rate of 30 frames-per-second, a model would have to be comprised of about 2,300 triangles. We will be viewing much larger models, on the order of 150,000 triangles or more. The limited update rate is the main limitation of this system and comes directly from the trade-off of price over performance. Given the size of data sets which must be viewed, the system is simply too

slow to display all the data. Therefore, a method which can reduce the amount of data while maintaining the useful information in the data is desired. To achieve the data reduction, a mesh reduction routine and a multiresolution mesh representation is implemented. These are discussed in the remaining chapters.

CHAPTER 3

Multiresolution Level-of-Detail Review

In order to visualize a set of range data, a set of polygon (triangle) meshes must be created from the captured data. The simplest polygon, the triangle, is also the most common polygon primitive used in computer graphics and all other polygons can be reduced to a set of triangles. Triangle rendering is supported by most graphics hardware due to it being the simplest representation possible and has thus remained a popular computer graphics primitive. The simplicity of the polygon representation does have its draw-backs, however. From a single 1024×1024 range image, a model containing over two million triangles can be generated. Merging data from 20-30 range images can produce models with up to 100 million triangles. This large number of polygons easily overwhelms even the fastest graphics machines available and presents a problem in the real time display of the meshes. Many of these meshes contain more data than is needed. For example, a flat surface maybe represented as hundreds of triangles instead of one polygon as shown in Figure *3.1*. Also, some older and smaller graphics systems simply can not handle even medium size models. Therefore, methods to reduce the number of polygons in a model are needed and the state-of-the-art methods are discussed in this chapter.

3.1 Terminology

Varying methods are available to represent the data structures used in mesh reduction and most methods have their own home brewed format, so more definitions will be given than will be actually used the implementation to give the background necessary to understand some of the algorithms. First, we wish to establish some terminology that will be used in this chapter starting with the most simple definitions to remove any ambiguities if possible. First, a *vertex* is of course a point in $\mathbb{R}^3$. Two vertices are connected by an *edge*. An edge can be internal to the resulting mesh, or it can be a boundary edge. Tying three or more non-linear, planar vertices together, and

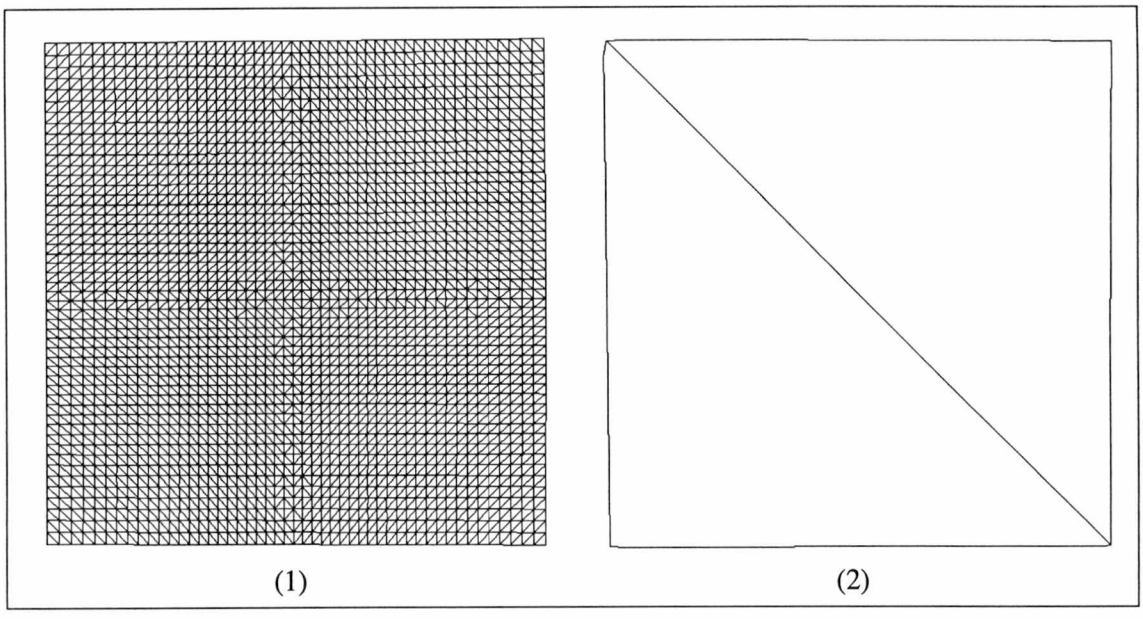


Figure 3.1: *(1) Triangle mesh created from a range image of a flat surface composed of 4802 polygons, and (2) the reduced geometry of the flat surface composed of only 2 triangles.*

thus 3 or more edges, into a closed sequence, or network, forms a polygon or *face*. A collection of connected faces is then known as a *polygon mesh*, or simply a mesh. A triangle is the simplest polygon that can be drawn and they are easy to manipulate. Many of the algorithms available only use triangle meshes since most hardware is optimized to draw triangles and all other polygons can be broken into triangles. (See Figure *3.2* for an example.)

Many of the algorithms use a manifold surface as their model, so we will discuss the definition of a surface and a manifold. A *manifold* defined for use in computer graphics is made up of many faces. It is a polygon mesh for which the neighborhood of every point is topologically equivalent to a disk. In other words, no vertex of the mesh lies on a boundary of the mesh, the mesh is closed. A manifold is therefore a water tight mesh. If the object is not water tight, it is a *manifold with boundary*. Therefore, in a manifold composed of triangles, each edge belongs to two triangles. For a manifold with boundary, each edge would belong to one or two triangles. A manifold is orientable if it has two sides which can be consistently labeled as side-one and side-two. A Möbius strip, for example, is non-orientable. A *surface* then is defined as a "compact, connected, orientable two-dimensional manifold, possibly with boundary, embedded in $\mathbb{R}^3$." If the surface

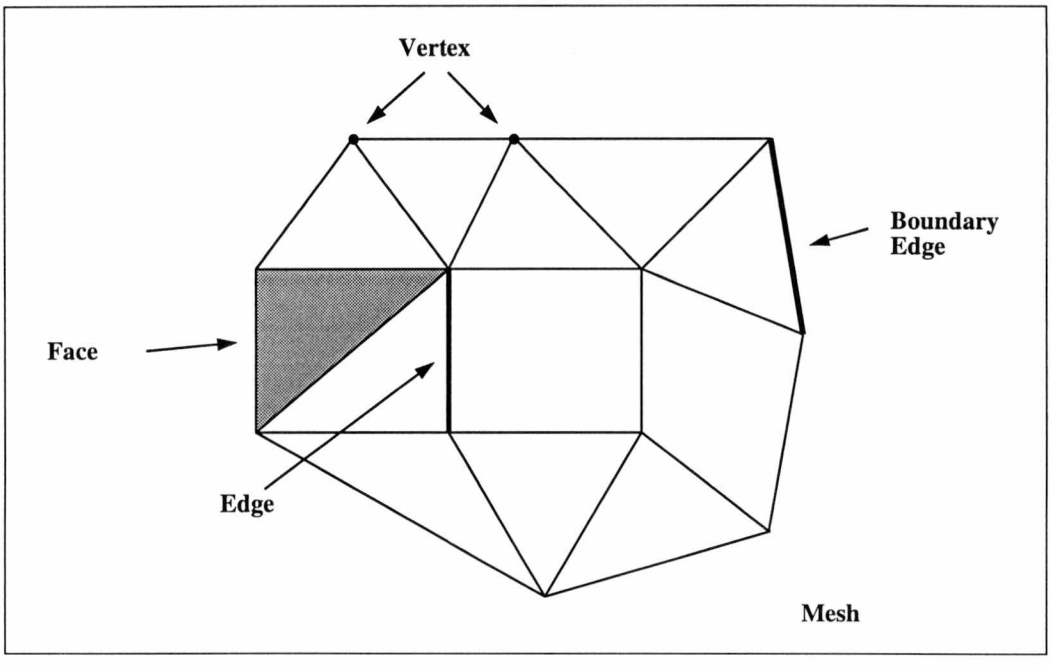


Figure 3.2: *Terminology used throughout this chapter.*

does not have a boundary, it is a closed surface. We will use the terms manifold and surface interchangeably.

Some algorithms use the mathematical idea of a simplicial complex in the representation of a triangle mesh. A simplicial complex S consists of a set $\{v\}$ of vertices and a set $\{s\}$ of finite nonempty subsets of $\{v\}$ called simplexes such that

- any set consisting of exactly one vertex is a simplex, and
- any nonempty subset of a simplex is a simplex. [97]

In this case, a *simplicial surface* is a piecewise linear surface with triangle faces. Building on the simplicial complex definition, a simplicial mesh M is defined as a pair (K, V) [52]. V is the list of vertices

$$\{v_1, v_2, ...v_m\}, \quad v_i \in I\!R^3. \tag{3.1}$$

(V) is the geometric realization in $I\!R^3$. K is the simplicial complex comprised of a vertex index

$$v \; = \; \{1, 2, ...m\} \tag{3.2}$$

an edge index

$$e = \{1,2\}, \ \{2,3\}, \ \ldots \tag{3.3}$$

and a face index

$$f = \{1,2,3\}, \ \{2,3,4\}, \ \ldots \tag{3.4}$$

$|K|$ is the topological realization in $\mathbf{e}^m$, the edge space. A simplicial neighborhood can then be defined for each of the simplices which is basically all the faces sharing a vertex with the simplex (see Figure *3.3*).

The creation of a polygon mesh can occur by many different means, such as marching cubes or zippered meshes. Creating a polygon mesh from a range image is a separate issue from the polygon reduction and we do not wish to dwell on mesh creation here. Our goal is to increase the speed at which a model is displayed. One method for accomplishing this is to use a *multiresolution model*, that is a model which contains continuous levels-of-detail of itself and can use any of the levels on demand. This method was first proposed by Clark [17] but has not been researched or implemented until recently due to recent advances in computer graphics hardware. The traditional level-of-detail implementation, such as the one built into the Open Inventor toolkit, consists of a few representations of a model to speed rendering. In a traditional level-of-detail representation each resolution model is stored separately (see Figure *3.4*). The model rendered is based on the distance from the view point. In this manner, lower resolution objects are drawn when the object is further away and details are not visible. The most recent release of the Open Inventor library (not available for SGI) contains a more continuous version of the level-of-detail which can be used based on other constraint besides distance, but again, it requires a separate model for each level-of-detail. A multiresolution model, on the other hand, consists of a much larger number of representation starting at one extreme, either the highest or lowest resolution model, and a means to continuously map to the other extreme. This method does not have the storage overhead associated with traditional level-of-detail since only one complete model is stored. Also, render criteria is not necessarily limited to a distance measure, but can be based on a number of viewing parameters, such as field-of-view or desired frame rate. One such multiresolution representation is the progressive mesh [49]. A progressive mesh is a continuous representation which contains

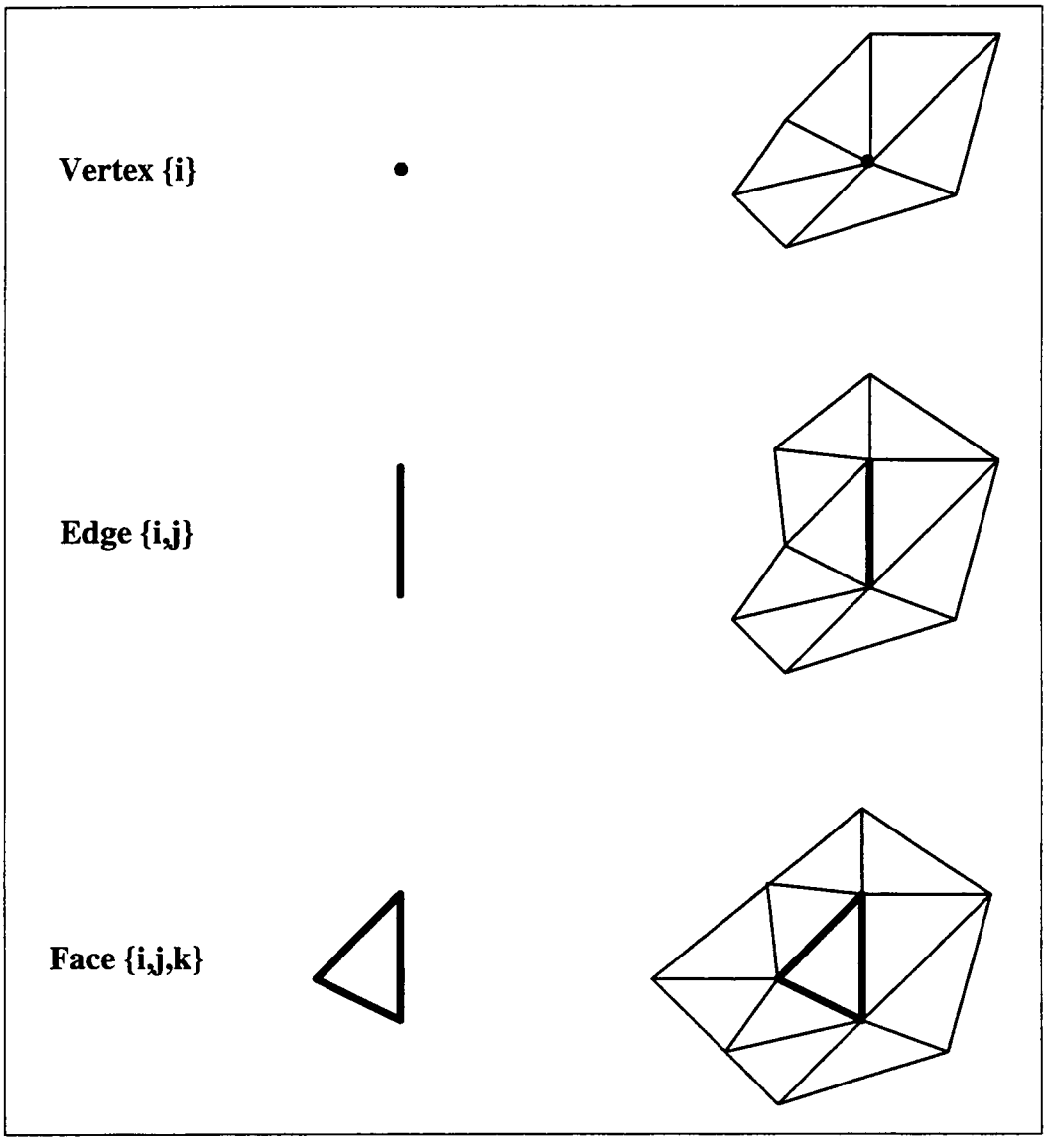


Figure 3.3: *Simplices and their simplicial neighborhoods.*

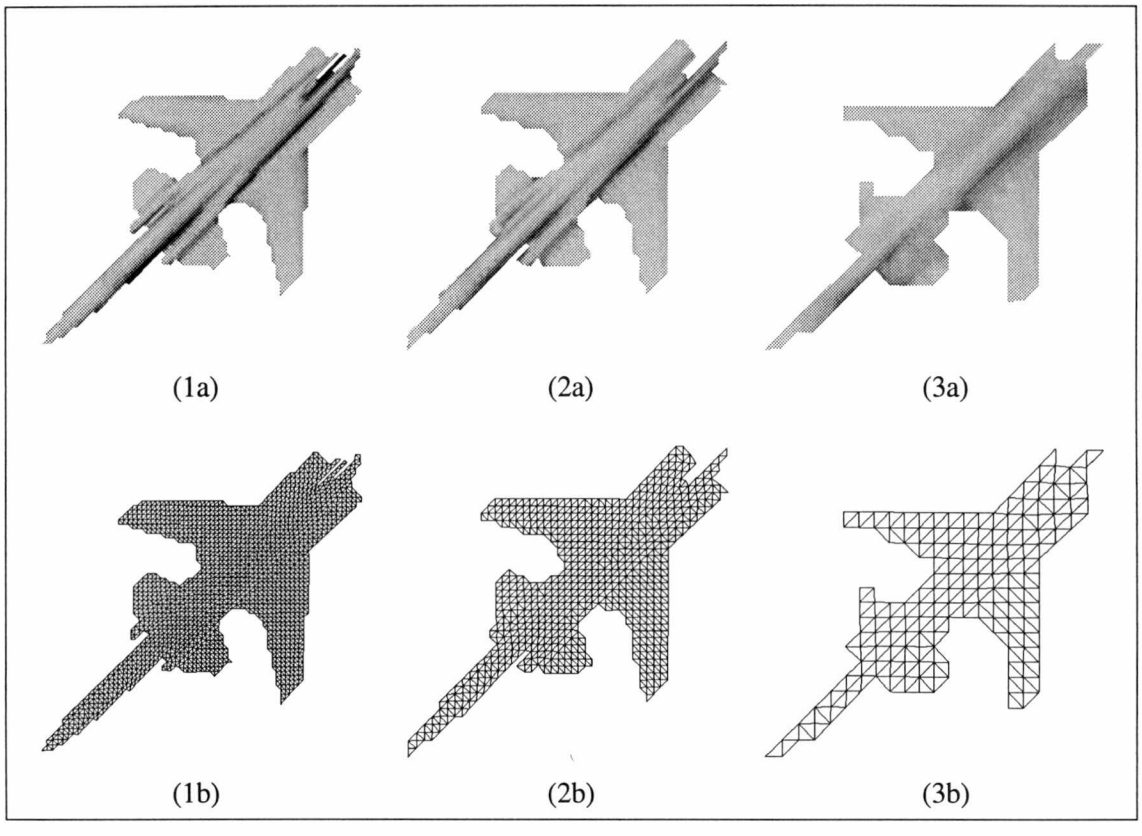


Figure 3.4: *Different level-of-detail models created using images pyramids of a synthetic range image of a plane with resolutions of (1) 90x90 pixels, (2) 60x60 pixels, and (3) 30x30 pixels.*

N versions of the model, M^0 through M^N, along with a mapping from each representation to the next. Other multiresolution models include octrees, quaternary triangulations, hierarchical representations, and wavelet representations.

Simplification is the method of automatically converting a detailed polygonal model into a simpler one. Simplification usually takes on one of two forms, *decimation* or *refinement*. Decimation takes a fully detailed model and creates simpler ones, while refinement takes a coarse model and adds details to create a finer model.

A *geomorph* is a smooth visual transition similar to 2D image morphing, but using a 3D model [49]. This term was first coined by Mark Kenworthy in 1992 to distinguish it from image morphing.

One unfortunate occurrence is that two definitions for *triangulation* are available in computer vision. The first is a method for recovering distance in systems such as structured light. Here, the second definition is used, that being a method for creating a triangle mesh from a set of vertices.

3.2 Overview of Polygon Reduction

Much of the research in polygon reduction and multiresolution modeling has taken place in the past few years due to the advent of computers which have the graphics capability to warrant such an effort. In the late 70's and early 80's, machines capable of several thousand polygons in real time first became available, but cost several million dollars [43]. Most multiresolution databases available today have been created by hand [32]. A few surveys and reviews of the young field have been written. Some of these are fairly complete reviews [78, 44, 84] while others only cover a few works [109, 94]. For example, Varshney [109] presents an overview of previous work in which he has been involved. There are several problem characteristics which each polygon model simplification algorithm must address including

- input/output attributes,
- data structure,
- error metric, and

- constraints.

The first issue to address is the topology and positional geometry of the input and output data. Basically, what type of data is input and output. This can include points, functions, height fields, surfaces, etc. In addition to the topology and geometry of the input, additional attributes, especially with multimodal data, can be used, such as color, texture, and surface normals. The domain of the output vertices must also be considered. They can be a subset of the input vertices or they can be resampled from the continuous domain. In some cases, the topology of the output may differ from the input, some methods preserve topology, others do not. Preserving topology is dependent on the application. For example, there is a performance hit when topology is preserved in both reduction and update rates, but perceived quality may be higher.

Structure of the data also effects the algorithm implementation. Some data sets are arranged on a grid, such as high field data. Others have a hierarchical representation. Some algorithms accept a general set of random vertices with no predetermined structure. The structure can be divided into one of three general categories[44]:

- height fields and parametric surfaces,
- manifold surfaces, and
- non-manifold surfaces.

The first category contains data sets which are height values on an (x, y) grid or can be described as a function of x and y: $H(x, y)$. These models have a natural 2D parameterization, and the reduction methods used for these do not apply directly to the 3D data with which we are dealing.

Two error metrics are typically used in mesh reduction algorithms, L_2 and L_∞ error. L_2 is the root squared error between two n-vectors $\mathbf{u}$ and $\mathbf{v}$ and is defined as

$$\|\mathbf{u} - \mathbf{v}\|_2 = \left[\sum_{i=1}^{n} (u_i - v_i)^2\right]^{\frac{1}{2}}. \tag{3.5}$$

The squared error is then defined to be the square of the L_2 error. The root mean square (RMS) error is the L_2 error divided by $\sqrt{n}$. L_∞ is the maximum error and is defined as

$$\|\mathbf{u} - \mathbf{v}\|_\infty = \max_{i=1}^{n} |u_i - v_i| \tag{3.6}$$

Optimization with respect to the L_2 and L_∞ error is known as least squares and minimax optimization respectively.

Constraints also factor into the selection of a mesh reduction algorithm. Limiting the computational time that a method has to reduce the mesh or the amount of error incurred while reducing the mesh are sometimes factors. Vertices that lie on boundaries and edges may have higher priority over internal vertices to help preserve visual quality.

There are also some characteristics associated with the simplification algorithms. The first is the speed/quality tradeoff. The fastest algorithms tend to have the lowest quality and those with higher quality tend to be slower. The algorithms can also be characterized by their starting point in the polygon reduction hierarchy. They can start with a minimal approximation and form models which are more and more accurate (refinement). The opposite approach is to start with the full polygon model and reduce the number of polygons to less and less accurate models (decimation). The number of passes through the algorithm also should be considered. Some algorithms require only one pass, where others require multiple passes.

Most algorithms input a set of polygons and produce another set of polygons, typically triangles. These methods can be divided into several distinct categories using the methodology employed by the algorithm. These are

- image pyramids,
- decimation methods (polygon merging, vertex decimation, edge contraction),
- hierarchical methods (volume methods, wavelet surfaces)
- vertex clustering,
- simplification envelopes, and
- retiling

Each method has its own advantages and disadvantages. For example, coplanar facet merging and decimation approaches tend to maintain sharp edges and angles better. Some of the methods rely

heavily on a strong mathematical basis, such as wavelet surfaces and mesh optimization. While others use simple geometric error.

This completes the discussion of how the algorithms are designed and categorized. Now a review of the more significant work in the area and the state-of-the-art methods is presented. The methods for mesh reduction can be divided into those which refine the mesh and those which decimate it. Our methods will focus mainly on the decimation techniques, but we will also review those for refinement to get a better overall view of mesh reduction in general. This will allow to better devise a plan as to how to approach the problem of reducing the multimodal data sets.

3.3 Height Fields

Much of the early work in triangle reduction was performed on height fields. Many of these methods take advantage of the inherent 2D grid layout of the data which eases the computational complexity of the problem. We will not focus on these methods here but rather methods for more general manifold surfaces and manifolds with boundary, the category into which our data fits. There is no natural 2D parameterization of manifold data, so they are more difficult to reduce than the height fields. At first, the laser range images that we are dealing with may seem to be similar to the area of height fields since both use a $2\frac{1}{2}$D data set aligned on a grid. The only difference being that the range image uses a (ρ, θ) spherical grid instead of an (x, y) planar grid. However, height field have a completely connected grid (no holes). Our data on the other hand is triangulated into an open surface with boundaries. Therefore, most of the methods used for height fields will not work with our more general 3D models. Also, using multiple sets of range data results in data that is no longer aligned along the same spherical grid.

Subsampled or hierarchical image pyramids are commonly used on height field data to produce different levels-of-detail [116]. Figure *3.5* gives an example of how an image pyramid is formed by subsampling an image. Neighboring pixels from the initial image are combined to form a pixel in the next level of the pyramid. This is continued until every level of the pyramid is formed. By creating an image pyramid from a range image, a level-of-detail model can be produced by creating a model from each image in the pyramid, but a multiresolution model can not be formed. Garland

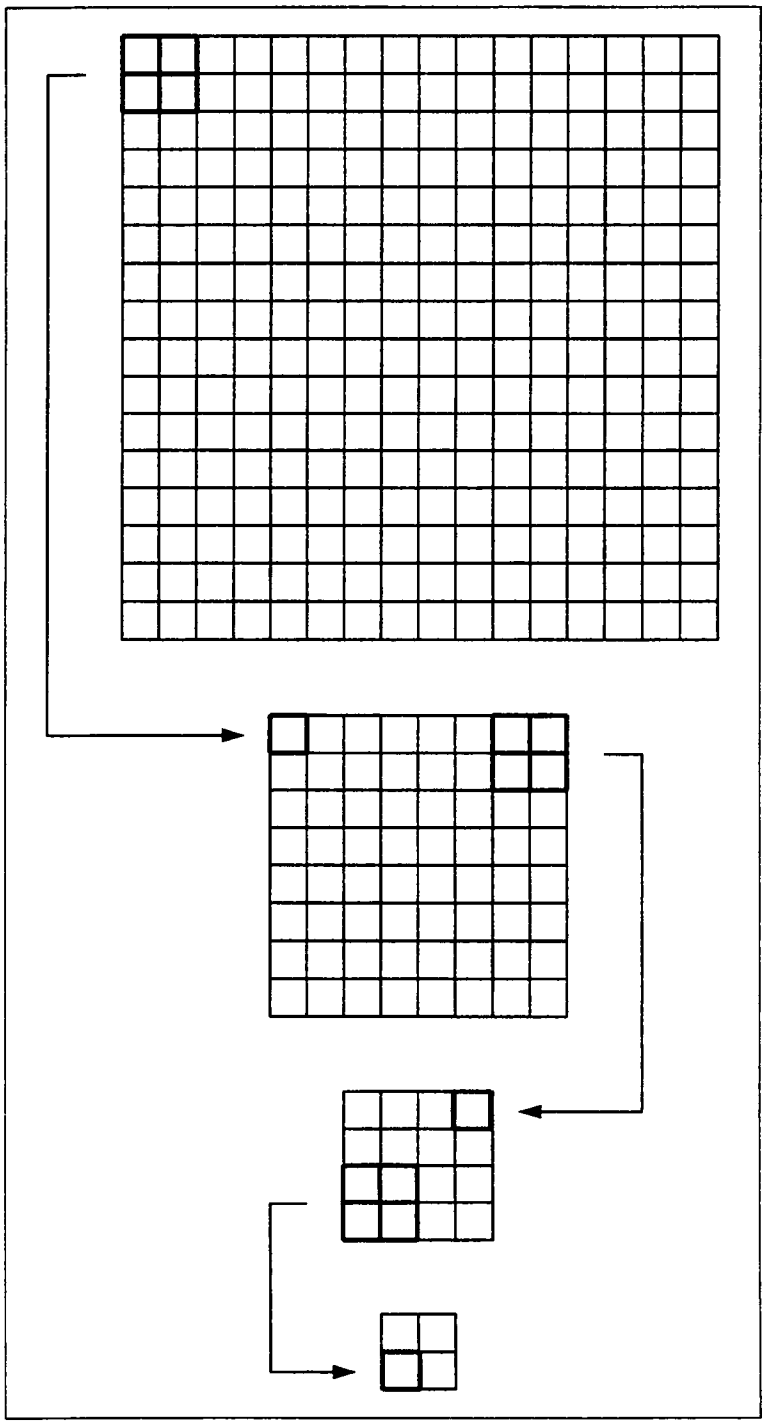


Figure 3.5: *Image pyramid creation.*

and Heckbert [34, 33] present an algorithm for approximating a height field using a triangulated surface which is piecewise-linear.

3.4 Manifold Methods

3.4.1 Manifold Refinement

Refinement methods start with a simple representation of the mesh and attempt to fill in details. Turk [106] introduces a simplification algorithms which inserts a number of vertices at random over a surface. The vertices are then distributed depending on the curvature. The original vertices are then iteratively removed, and the surface is re-triangulated to get the final mesh. This algorithm is best suited for models that are meant to represent curved surfaces. This method creates several levels-of-detail of a given model. Zorin *et al.* [120] present a method which is aimed toward editing a mesh representation of a model for animation or special effects. This method uses triangle subdivision to create a multiresolution mesh in which each triangle is subdivided into four triangles.

3.4.2 Manifold Decimation

Decimation methods remove vertices, edges, triangles, or entire patches of triangles. This is the area in which we will focus most of our attention. This is due to the nature of the data which we will be processing. The data set is obtained from a laser range scanner. This information tends to be dense and redundant which lends itself to decimation. Triangulation can be quickly implemented on this data set due to its layout [107].

3.4.3 Coplanar Facet Merging

Coplanar facet merging involves joining adjacent polygons which have the same, or nearly the same, normals. These methods are very quick and maintain sharp edges and angles very well. An example is shown in Figure *3.6* were a plane is represented by by a mesh containing 36 triangles. By merging the coplanar faces, removing colinear points, and retriangulating the area, the mesh is reduced to 2 triangles. Kalvin *et al.* [55] describe an algorithm which first approximates a

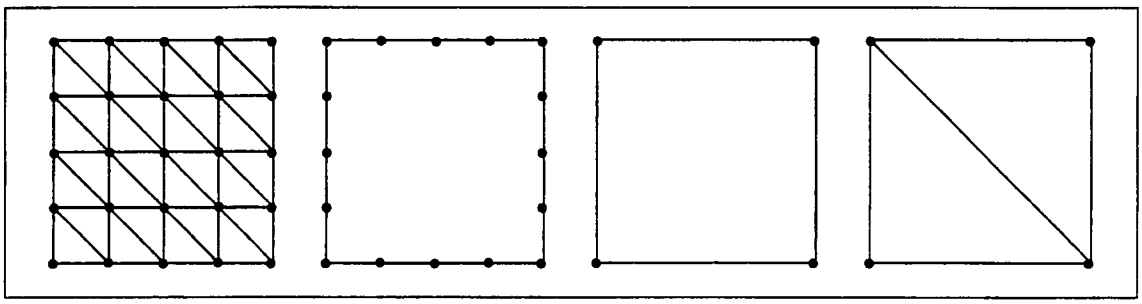


Figure 3.6: *Coplanar facet merging.*

surface with polygons using a method similar to marching cubes. The model is then decimated by merging adjacent faces which are precisely coplanar.

Hinker and Hansen [46] present a rather intuitive method for mesh reduction. This is a one pass method which finds patches of triangles with nearly parallel normal vectors and retriangulates each patch. This method is stated to be "largely ineffective when faced with surfaces of high curvature."

Kalvin and Taylor [56] present their method called Superfaces, a general purpose algorithm for simplifying polyhedral meshes. This method applies to polyhedral meshes that need not have triangle faces and uses a bounded approximation approach which guarantees that every vertex in the original mesh lies within a user specified distance of the simplified mesh. The algorithm works by first forming a superface, that is a nonplanar polygon formed by the boundary of a surface patch. These patches are chosen from quasilinear triangles that satisfy the error criterion, which is an improvement in reduction over Hinker's [46] method which merges only coplanar triangles. The borders of these superfaces are then straightened, and the resulting superface retriangulated. This method preserves the topology of the original mesh and the resulting vertices form a proper subset of the original vertices. It does, however, only produce specific levels-of-detail. Also, the resulting meshes can have intersecting triangles. Kalvin and Taylor do present a comparison of other mesh reduction algorithms, but do state that getting meaningful results from this comparison is not simple because there is no consistency in the data sets or the hardware from one method to the next.

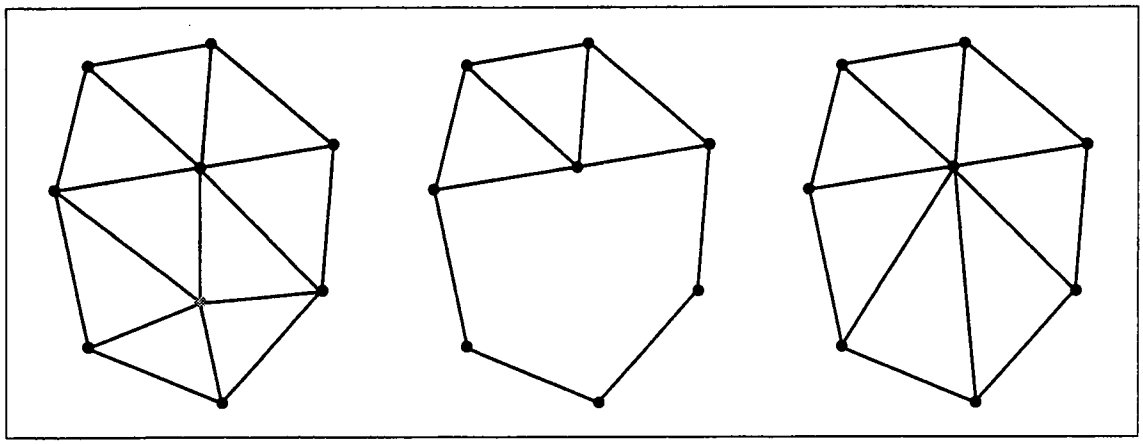


Figure 3.7: *Vertex decimation.*

3.4.4 Vertex Decimation

Vertex decimation methods are iterative algorithms which remove vertices and retriangulate the resulting region. Figure *3.7* shows an example of this. Here a single vertex is removed deleting five surrounding triangles. The resulting hole is then retriangulated forming three new triangles. Schroeder *et al.* [92] introduce a method for general polygon mesh decimation that uses local operations to reduce the number of triangles. This method preserved the topology of the mesh while approximating the geometry. This is a multiple pass method. The first step of each pass is to characterize the geometry and topology of each vertex. Each vertex falls into one of five categories: simple, complex, boundary, interior edge, or corner. All vertices except those classified as complex are candidates for removal based on the decimation criteria. Simple vertices use the distance to plane criterion. Boundary and interior edge vertices use the distance to edge criterion. If the vertex is within a specified distance to the plane or edge, it is deleted. The resulting hole left by the removal of the vertex is then retriangulated. Removal of simple, corner, or interior edge vertices reduce the mesh by two triangles, and removal of boundary vertices reduced the mesh by one triangle. The original algorithm measures error based on the previous approximation, not the original input, so errors can accumulate with each pass. This method is fairly fast and produces moderate quality results. This method does not produce optimal results since multiple

vertices can be deleted on each pass. Schroeder [89] presents an algorithm based on earlier work [92] which incorporates the progressive mesh (PM) representation. This method also modifies the topology of the model to achieve greater reduction, where most methods have strived to maintain the topology. Schroeder introduces another pair of operators, vertex split/merge, to accompany the edge collapse/split from Hoppe [49]. This algorithm again classifies all the vertices and computes an error value based on this classification as in [92]. Retriangulation is then performed by edge collapse instead of vertex deletion. If the desired reduction has not been met, the mesh is then split and the process continues. This method does not maintain a fixed error, but instead concludes when a desired reduction is achieved. Schroeder and Citriniti [90] present the decimation algorithm that is implemented in their Visualization Toolkit [88]. This is the same algorithm presented earlier in [92] with some recent extensions into VRML.

Ciampalini *et al.* [15] present a new mesh decimation method called JADE (Just Another DEcimation). This algorithm is based on Schroeder's decimation [92], but instead of a local error measure, the decimation is based on a global error. The other main goals of this algorithm were to have a low memory overhead and to provide a real multiresolution representation. The JADE multiresolution mesh representation uses an interval tree for storage, and as such, does not allow geomorphs. Also, no scalar quantities are represented in this method. The output of the simplification is a subset of the original vertices.

Soucy [94] presents the multiresolution modeling algorithms used at InnovMetric which is based on earlier work [95, 96]. This is a vertex decimation method based on a sequential optimization process which preserves local topology and surface edges. Mapping of RGB colors onto the reduced models is incorporated. Also, the maximum error is bounded. However, no details of the method are given.

3.4.5 Edge Contraction and Mesh Optimization

Edge contraction replaces an edge with a single vertex by moving the two vertices forming the edge to one position. These methods produce high quality results and support continuous level-of-detail models. Figure *3.8* shows an example of edge contraction. Here, an edge is selected for

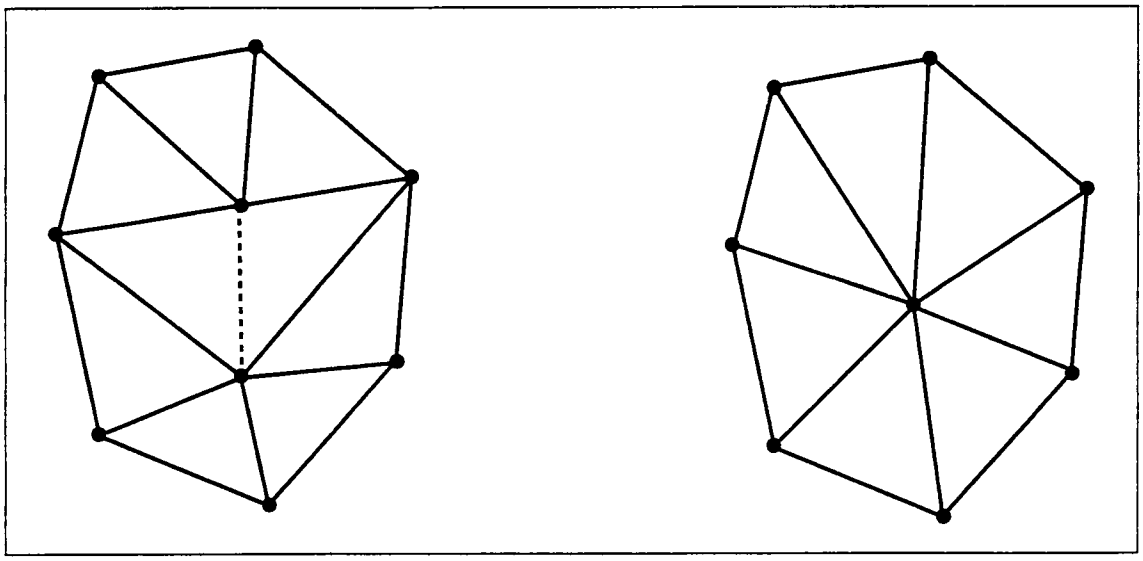


Figure 3.8: *Edge contraction.*

contraction. The two vertices at the ends of the edge are effectively merged together into one new vertex. This removes the two faces which were comprised of the edge and reshapes the neighboring faces. Hoppe *et al.* [52, 48] introduces a method for mesh optimization which employs an energy function which attempts to measure the quality of the mesh. The mesh optimization algorithm uses edge contraction as a means to reduce the mesh once a candidate edge is chosen by optimizing an energy function. This algorithm, given a set of data points and an initial triangle mesh, produces a mesh of the same topological type as the input mesh that fits the data well and has a small number of vertices. The number, connectivity, and position of the vertices are changed to produce the reduced mesh by optimizing an energy function which measures the deviation of the final mesh from the initial mesh. The energy function is composed of three terms: one for distance from the final surface to the initial points

$$E_{dist}(K, V) = \sum_{i=1}^{n} d^2(\mathbf{x}_i, \phi_v(|K|)), \tag{3.7}$$

one for number of vertices

$$E_{rep}(K) = c_{rep} m, \tag{3.8}$$

and one to minimize long triangles during triangulation

$$E_{spring}(K,V) = \sum_{\{j,k\}\in K} \kappa \, ||\mathbf{v}_j - \mathbf{v}_k||^2. \tag{3.9}$$

This gives an energy function

$$E(K,V) \;=\; E_{dist}(K,V) + E_{rep}(K) + E_{spring}(K,V). \tag{3.10}$$

to minimize over the set of simplicial complexes. This method tries to retain vertices in areas with higher surface gradients while aligning edges along directions of less curved surfaces. This method employs an L_2 norm, but uses a global error approach. It also adds new points along with removing the older one to try to arrive at the optimal solution. The algorithm is driven by a set of tuning parameters set by the user which have been described as "hard to interpret" [78]. The result are high quality, but at the cost of very long processing times.

Hoppe [49] introduces the idea of the progressive mesh (PM) representation of polygonal meshes. This method stores a continuous resolution representation of the mesh, instead of a few levels-of-detail that is generated by other methods. In this representation, a mesh $\hat{M}$ is stored as a reduced mesh M^0 with a sequence of n records to refine M^0 back exactly to $\hat{M} = M^n$. Here, Hoppe refines some of the methods from earlier work [52, 48] to a simpler algorithm. Edge collapse is sufficient for effectively simplifying meshes, so edge split and edge swap are not needed to create the PM. This method also tries to preserve scalar attributes, such as color and normals, by adding an additional term to the energy function

$$E_{scalar}(V) = (c_{scalar})^2 \sum_i ||\mathbf{x}_i - \phi_v(\mathbf{b}_i)||^2. \tag{3.11}$$

Also, the term E_{disc} is added to the energy equation. This term is defined like E_{dist} except the points are constrained to a set of sharp edges to try to preserve discontinuity curves.

As a result of storing the sequence of edge collapses, or vertex splits, less storage is needed than storing multiple levels-of-detail. Another result is that geomorphs can be created between any two meshes of a PM. The PM also supports selective refinement in which detail is added to the model only in desired areas. This is achieved by not performing a vertex split if the area is not in

the area to be refined. This method appears to be the most useful representation of level-of-detail meshes. However, it currently only applies to 2D manifolds and is rather complex.

Hoppe [50] introduces a real-time framework for selectively refining a progressive mesh based on the view frustum, surface orientation, and screen-space geometric error. This is accomplished by creating new vertex split and edge collapse routines and establishing a parent-child relation on the vertices. Vertex splitting to refine the PM is then performed based on the view frustum, surface orientation, and screen-space geometry.

Garland and Heckbert [35] present an algorithm which iteratively contracts edge pairs using quadric error metrics. This method is similar to vertex clustering. The error is stored in an error quadric matrix which is efficient for evaluating total error. This method does not preserve topology, but allows simplification of non-manifold and disconnected meshes.

3.4.6 Volume Methods

Volumetric methods are octree or voxel based. These methods apply a sampling approach, such as a multiresolution marching cubes, to produce varying levels-of-detail. They generally do not preserve topology or sharp features of the model. He *et al.* [41] presents a simple method for object simplification which takes a signal processing approach to gradually eliminate high frequency details in a model. This method uses a voxel based approach which volume samples and low-pass filters the object into multiple resolution volumes buffers (a 3D pyramid of raster volumes). The marching cubes algorithm is then used to generate the new models. This method does not preserve topology or edges and only works on closed surfaces. Its main use is for more traditional LOD models which use lower levels-of-detail at greater distances when details can not be seen. The output of this method is produced by the marching cubes algorithm, which could produce yet another large mesh.

He *et al.* [42] present a rework of [41] using an adaptive marching cubes algorithm. This adaptive method preserves the topology and guarantees a user specified error bound of the mesh generated by the standard marching cubes. This algorithm, however, still only works for closed surfaces. It is also slow, and accurate, compact models are not easily produced.

Renze and Oliver [80] focus their method on tetrahedral volume decimation. This method does not currently create a multiresolution surface and global properties such as maximum node displacement are not constrained.

Luebke [65] presents a hierarchical dynamic simplification method based on an octree data structure. The octree contains information about the vertices and triangles of the model. This is a vertex clustering method which does not preserve topology. Node collapse and expansion is based on their screen-space area and retessilation occurs continuously. Luebke and Erikson [66] present a revision of their hierarchical dynamic simplification method [65]. The octree notation is dropped in favor of a vertex tree hierarchy. Luebke states that this method runs adequately on small models, but larger models present problems. This method was designed to deal with real-world CAD models which are topologically unsound.

Xia *et al.* [118] present a method which is similar to the PM [49], but instead of a sequence of edge collapse/split, constructs a merge tree over the vertices which stores the edge collapses in a hierarchical manner. This method also uses image space feedback to more intelligently create the level-of-detail selection so as to reduce the amount of popping visible by the user. This allows the level-of-detail to be selected at run-time based on the image space of the model after the merge tree has been created off-line. Local illumination and visibility culling are also performed during run-time to increase performance. Computing the level-of-detail during run-time does have the disadvantage of trading display time for computational time. This method was tested on a Silicon Graphics Onyx RE2 with a 194MHz R10000 processor and 640 MB RAM and gave results varying from 40,000 - 60,000 triangles per second which gives 1000-2000 triangles at frame rate.

3.4.7 Simplification Envelopes

Simplification envelopes give a way to guarantee error bounds. They work by creating an inner and outer envelope of the model. A new model is then constructed inside the two resulting envelopes. Figure *3.9* gives a two-dimensional example of this concept. Here, inner and outer offset surfaces are formed around the object. A new object is then drawn within the bounds of the offset surfaces using fewer vertices resulting in a reduced model. Cohen *et al.* [18] present the idea of

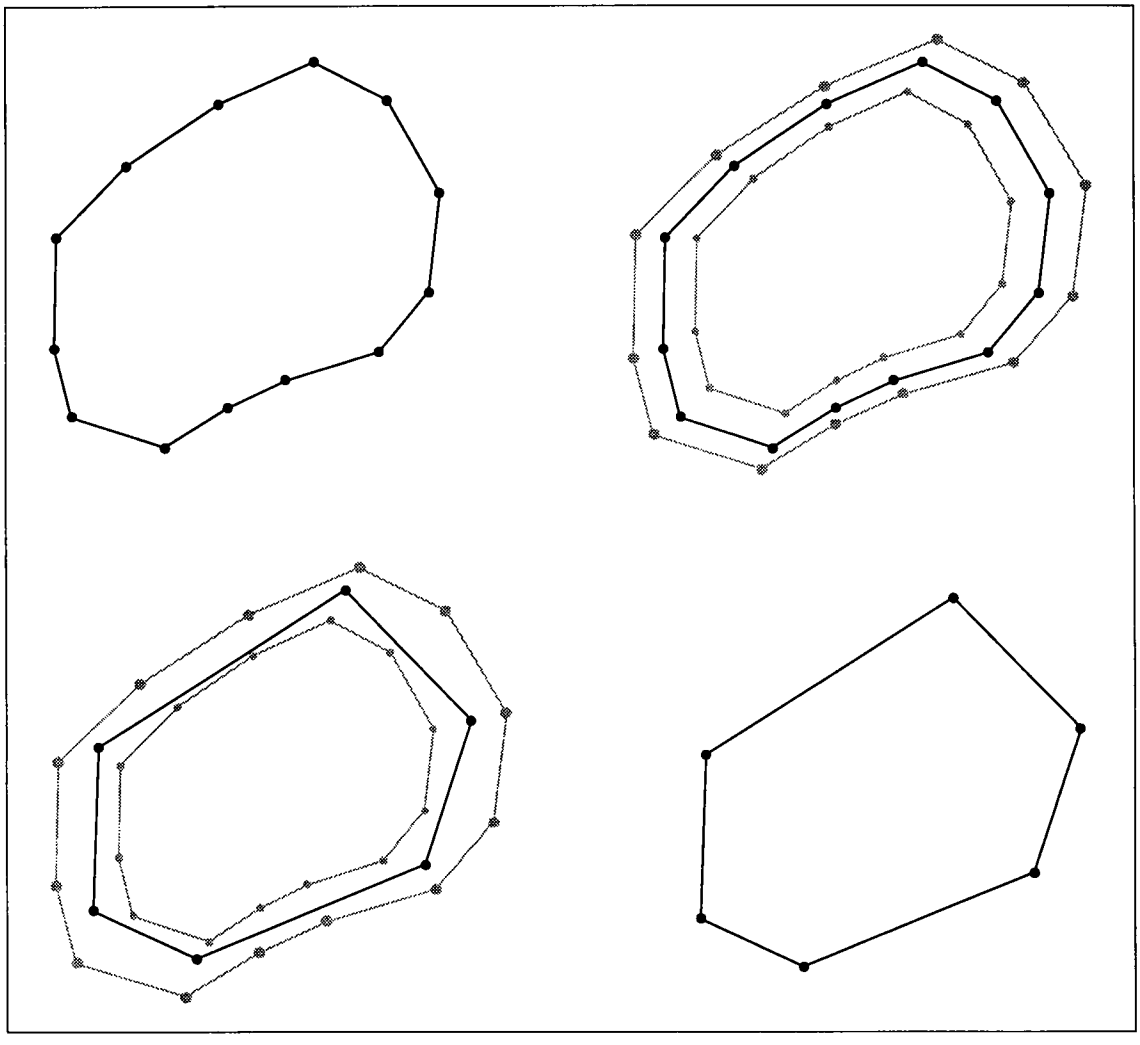


Figure 3.9: *Simplification envelopes concept performed on a two-dimensional model.*

simplification envelopes for generating a hierarchy of level-of-detail models. The user sets an error bound ϵ which is used to limit the outer and inner envelope's distance from the original mesh. The algorithm only works with manifold triangle meshes, with border edges forming a special case in the reduction scheme. The method is slow ($O(n^4)$), so it is very inefficient for large data sets. The resulting data set is a subset of the original vertices.

3.4.8 Wavelet Surfaces

Wavelet modeling of a mesh is not simple. These methods require that the surface be remeshed to support regular subdivision connectivity which may result in error being introduced into the high level details of the model. These methods are lossy, being unable to preserve high frequencies such as edges and discontinuities, and are difficult to implement. The wavelet model is, however, inherently multiresolution. DeRose *et al.* [23] introduced an algorithm to simplify polygons by representing the surface as wavelets. Eck *et al.* [25] presents extensions to earlier work in mesh reduction using wavelets [23].

3.4.9 Others

These papers do not fit into any of the previous categories, but they are, however, relevant to mesh reduction. Therefore, they are included here. Hoppe *et al.* [51, 22] presents a method for computing a surface from an unorganized set of points. The algorithm automatically infers the topology of the surface based on the idea of determining the zero set of an estimated signed distance function. These surfaces are used as the input to a mesh optimization algorithm (see Hoppe 93). This method was later improved [50]. Pulli *et al.* [77] also present a refinement of the earlier work [51] in mesh creation from a set of vertices.

Schroeder and Yamrom [91] present a data structure for efficient visualization algorithms. This structure is a variation of display lists with additional hierarchical information. It provides the adjacency information available in an hierarchical representation, not available in simple display lists, without the large amounts of memory required for an elaborated hierarchical structure.

Varshney *et al.* [108, 110] present an algorithm for generating various levels-of-detail for a polygonal model which guarantees that all the points of the approximation are within a user-

specified distance ϵ from the original model and that all the points of the original model are within ϵ from the approximation. This is accomplished by using a set partition problem. The algorithm preserves topology and can be constrained to preserve user-specified edges. It assumes that all polygons are triangles and are well behaved. This method produces a few specific levels-of-detail by simplifying the previous level. In this manner, error can accumulate as the hierarchy is built.

Evans *et al.* [29] present a method for generating triangle strips. This method introduces a "swap" command into the triangle strip which allows for longer strips to be generated than traditional methods. Triangle strips can be produced from a triangle mesh after the mesh has been reduced to further increase the frame rates.

Cignoni *et al.* [16] present a unique approach to multiresolution representation of polygonal meshes. All techniques surveyed thus far have taken a mesh as input to the algorithm and output a mesh with a reduced representation. This method, however, produces the reduced mesh by reducing the input data. The mesh is then constructed from this reduced data set.

3.5 Non-Manifold Methods

Popović and Hoppe [75] introduce the progressive simplicial complex representation for triangulated models. This representation is a generalization of the progressive mesh representation which can only be used for 2D manifolds. This method works with triangulations which are of any dimension, are non-orientable, are non-manifold, or are non-regular. This is accomplished by using a different simplex to represent the model. A generalized vertex split is also added to take into account this new data structure and the edge collapse is replaced by a vertex unification routine. However, this method only works with triangle data.

3.5.1 Vertex Clustering

Vertex clustering methods involve replacing a group of closely located vertices with one vertex. These methods are quick, but have low quality results. Topology is not preserved using these techniques. Figure *3.10* gives an example using a two-dimensional model. Here the model is sectioned on a 2D grid (a 3D volume would be used in the case of a three-dimensional model). All

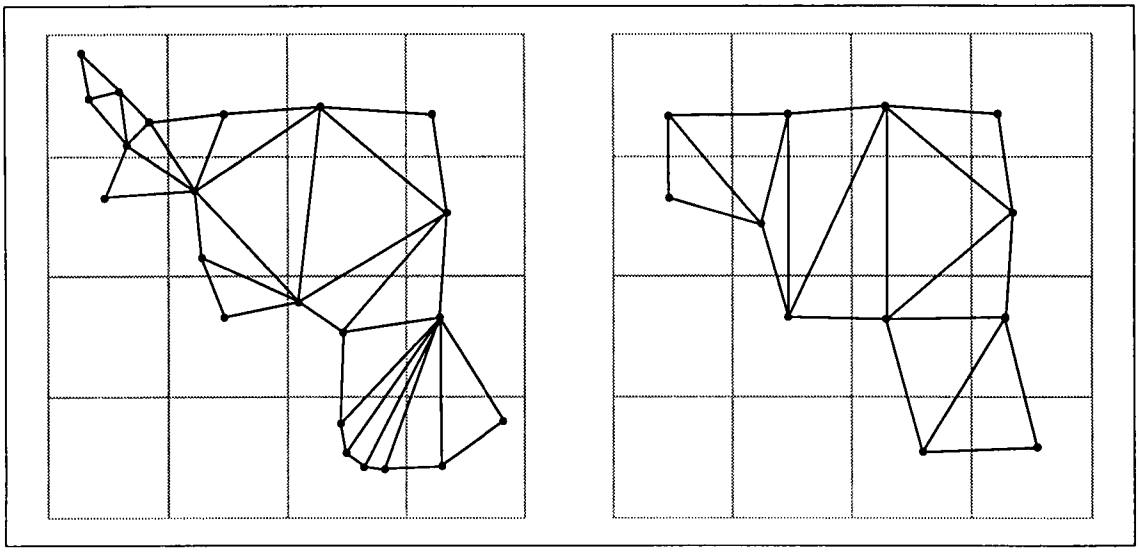


Figure 3.10: *Vertex clustering on a two-dimensional model.*

the vertices within a single grid are replaced by a single vertex. The model is then retriangulated based using the new vertices forming reduced model. Rossignac and Borrel [85, 86] attempt to simplify general, polyhedral models with arbitrary topology, where most others have dealt with triangle meshes. They approach simplification as a signal processing problem, the model is filtered, resampled, and reconstructed. This is accomplished by clustering vertices that are close to one another to create a new vertex at the center of mass of the cluster. This gives rise to problems which are common in signal processing. Aliasing can arise and high frequency details can be removed.

3.6 Conclusions

As can be seen by the wide variety of methods presented in this chapter, there is no standard reduction method, or even mesh representation. There are several desired qualities for the reduction method used for the data sets we are visualizing. The method used for reduction must be able to handle the multimodal data sets we have. A vast majority of the current methods do not. It must also be easily extendable to include any future data modalities that may be used. It is also very desirable to use a common data format for the models so the models can be easily exchanged

with other people and used on different computer platforms. All of the current methods use an in-house format. In addition, we want the reduction method to be capable of output a multiresolution representation of the reduced model.

Many different techniques are available for the reduction of polygon meshes. The papers reviewed which present the state-of-the-art approaches for mesh reduction include Turk's retiling [106], Schroeder's decimation method [92], Hoppe's mesh optimization [52], Hinker's coplanar facet merging [46], Eck's wavelet surfaces [25], Cohen's simplification envelopes [18], and Hoppe's progressive meshes [49]. However, none of these methods alone have all the desired characteristics which we wish to have in a reduction method. Hoppe [49] has the best approach for representing a multiresolution or continuous level-of-detail model. The PM approach stores in compact fashion all the information needed to make any desired resolution of a model. The mesh optimization algorithm, however, is not easily extendable to incorporate data from another modality. Hinker and Hanson [46] on the other hand have a method that is very quick but doesn't achieve the same quality of reduction. Current methods are fast or high quality or multiresolution, but none possess all of these qualities and none are easily extendable to include other modalities. Other characteristics that we desire include using a fine-to-coarse decimation method due to the nature of our data, which rules out retiling. We also do not wish to rely on heavily mathematical algorithms which are difficult to extend to include other modalities, which rules out the wavelet and optimization approaches. Also, most methods also only consider geometric shape in the reduction, not color, texture, etc. that we have with the multimodal data. We wish to set back and take a look at the bigger picture of what needs to be accomplished. We want to reduce the number of triangles, but we have to decide what criteria we wish to meet with the reduction. One issue with which we are faced is the fact that we have many objects in the range images we are reducing. Most of the current techniques for generating multiresolution models are optimized for small numbers objects with a large number of polygons [109]. Most of the methods reviewed which reconstruct range images only have one object in range images. Also, many of the algorithms only work with manifold objects. We do not want to limit our method in this way.

Therefore, the reduction method which we desire is a decimation method (fine-to-coarse) ca-

pable of handling multimodal data of multiple objects and outputting a multiresolution mesh. The reduction should be based on a variety of features of the model, including geometry, shape, and color. Furthermore, the method should be easily extendable to include any future modality of data. Since each of the modalities is a feature associated with the model, we propose a feature based reduction method. The following chapter describes in more detail the new method which is created based on a pattern vector representation of the data.

CHAPTER 4

Pattern Vector Based Mesh Reduction and Multiresolution Representation

Not forgetting the big picture, the high level goal of this research is visualization. So far we have reviewed the relevant literature and discussed the design and setup of the visualization hardware. The heart of this research, however, lies in the development of a novel triangle mesh reduction technique which is multiresolution capable. This chapter gives the details of the implementation with the results given in the following chapter.

4.1 Goal

The goal of this research is to design and build a system for use in the visualization of multimodal range data sets. A methodology utilizing the latest virtual reality hardware has been used in implementation, along with object oriented programming and the use of common toolkits to keep the code portable and modular. Also, keeping the cost of the hardware low has been a primary concern during the design and development of the hardware portion of the system. Specific steps completed to realize this system are as follows:

- create texture-mapped models from registered pairs of range and intensity images,
- interact with the models using VR hardware,
- build a large screen, multiple user, interactive CAVE system,
- develop a method to create multiresolution models, and
- the real-time display of the models created.

The result is a system which has a simple high level interface making it easy to use. The design of the large screen projection system has already been described in Section *2.4*. The following

sections address the main research issue – *a new technique for the reduction of multimodal 3D meshes.*

4.2 Reduction Methodology

The previous chapter reviewed the current state-of-the-art reduction methodologies and techniques. Also presented was the idea of a feature based reduction algorithm. With this in mind, we can apply techniques developed for pattern recognition [102], where in this case, our pattern or feature vector is comprised of features of the model to be reduced. The reduction method presented here is based on the edge collapse/vertex split concept since this method lends itself easily to the multiresolution mesh concept. The uniqueness of this reduction method, however, comes in the form of the data representation. We not only use the position of the vertices, but also other geometric properties such as normal and curvature information which can be derived from the edge connections as described in Section *4.2.1*. In addition, we have the advantage of having even more information about the mesh than just structural information. One of these comes in the form of grey-scale and/or color intensity images. We use this multimodal information to our advantage and have designed a triangle mesh reduction algorithm which uses this extra data. This is accomplished by mapping the data into a higher dimensional pattern space in which each vertex and its properties are represented by a pattern or feature vector. Each vertex is assigned an n-dimension pattern vector based on position, color, type (boundary, interior, etc.), and any other data feature available. Each vector also has an associated weight. Using the initial network of edge connections and the newly created feature space, the order of the edge collapses is determined using all the given information. This is accomplished by calculating edge lengths in the higher dimensional space using the euclidean distance between the connected vertices in the feature space. The order of the edge collapses is determined by sorting the calculated edge lengths. Edge collapse is then performed based on distance of the points in $\mathbb{R}^n$. The resulting order of collapse is stored in a dendrogram tree structure. The tree structure will also satisfy the multiresolution aspect of the project since the tree can be traversed as far as needed to recover the desired resolution of the model. Figure *4.1* shows an example of part of a dendrogram tree. Some of the initial vertices are

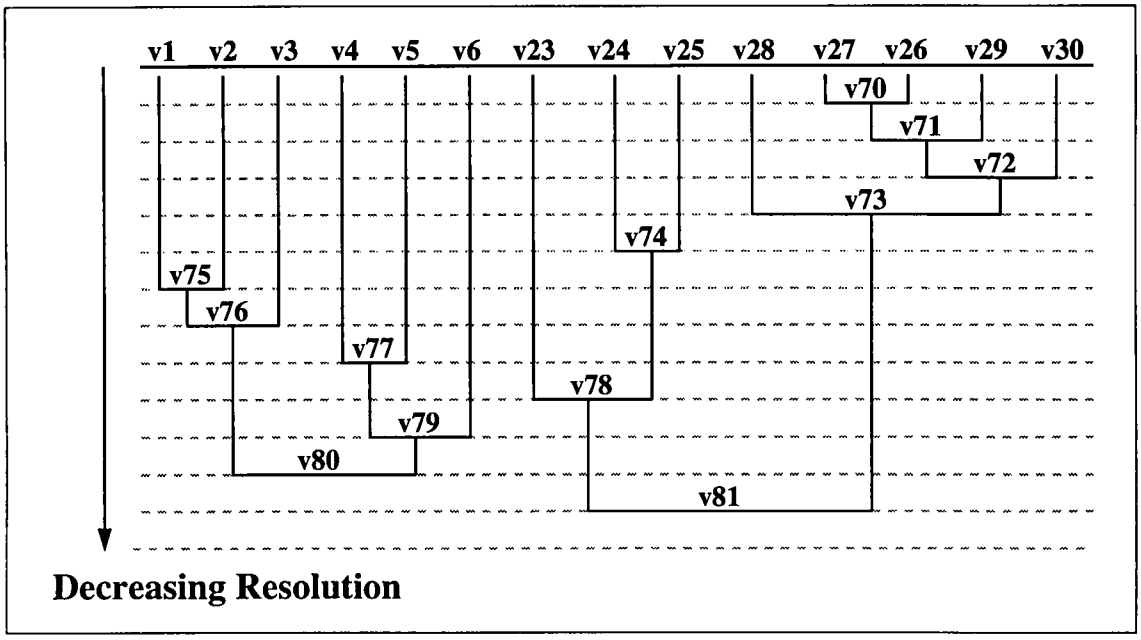


Figure 4.1: *Dendrogram tree structure used to represent the multiresolution mesh created from the edge collapse reduction.*

given at the top of the tree. At each level down the tree a single edge collapse is performed creating a new vertex and reducing the mesh by one or two triangles. Looking across the dendrogram gives all the vertices used for a given level or resolution in the representation. Performing the reduction in this method meets our criteria of basing the polygon reduction on more than simple geometry as most current methods do. Also, the vectors can be weighted to prefer one feature over another. If, for example, the color of the object is a more important feature to retain, then the color weight can be set higher.

A problem for implementing any mesh reduction technique is that there are no standards in place for 3D data reduction. That is, there are no measures by which one method can be compared to another. All the algorithms reviewed have been implemented using in-house data formats. A common method for measuring quality of output meshes or the associated error is not available. Reduction time appears in most papers for a given method on a given hardware setup, but there is no common benchmark to compare the results of differing techniques. We wish to at least have hooks to allow the use of common toolkits for a standard data format. Unfortunately, we do

not have a solution for testing one algorithm against another, but do give results of our method, including error analysis and timing, in the following chapter.

4.2.1 Pattern Vectors

Several factors have been included in the feature vectors used to determine edge length in the reduction method. These include:

- geometry,
- color,
- normal,
- curvature, and
- type.

The *geometry* is of course the (x, y, z) position of the vertex. The *color* is an RGB triplet containing the diffuse color or intensity of the object at that vertex. The *normal* of a vertex, V_i, is calculated by averaging the normals of all the faces which contain that vertex (see Appendix *A.4*):

$$\vec{N}_i = \frac{\sum_{j=1}^{n} \vec{F}_j}{n} \tag{4.1}$$

where $\vec{N}_i$ is the normal calculated at vertex V_i, $\vec{F}_j$ is the normal of a face which contains V_i, and n is the number of faces containing V_i. The curvature for a vertex in a model can be defined in a variety of ways. Here we use a type of local dispersion of the normals. The *curvature*, C_i, at a vertex, V_i, is therefore calculated as the variance of the normal:

$$C_i = \sqrt{\frac{\sum_{j=1}^{n} \left|\vec{F}_j - \vec{N}_i\right|^2}{n}} \tag{4.2}$$

The *type* feature currently is simply set to external or internal depending if the vertex is on a boundary. This is usefully in preserving the boundaries of objects. Other features can be quickly and easily added as needed by simply increasing the number of dimensions in the feature space.

These could include thermal or radiation information. Each of the vectors also has an associated weight. A higher weight for a feature will allow it to be biased over another.

A simple example of how the new feature space is used to calculate the order of the edge collapse is shown in Figure *4.2*. Here a simple mesh with 16 vertices in $\mathbb{R}^2$ is shown. Since it is not possible to draw a picture in more than $\mathbb{R}^3$, this example starts with a mesh in $\mathbb{R}^2$. The vertices and the edge network are then mapped into $\mathbb{R}^3$ using the boundary information for the extra dimension. In general, the lengths of the edges are then calculated using a simple norm in the higher dimensional feature space between the weighted vectors derived from the vertices which an edge connects. That is:

$$\begin{aligned} L_{ij} &= |\vec{w}_i\vec{v}_i - \vec{w}_j\vec{v}_j| \\ &= \vec{w}_{ij}\,|\vec{v}_i - \vec{v}_j| \end{aligned} \tag{4.3}$$

where L_{ij} is the calculated edge length between the two pattern vectors $\vec{v}_i$ and $\vec{v}_j$ derived from the vertices V_i and V_j, respectively, along with the associated weight vectors $\vec{w}_i$ and $\vec{w}_j$. Throughout the use in this dissertation, $\vec{w}_i = \vec{w}_j = \vec{w}_{ij}$. Therefore, in the example, when edge lengths are calculated in the higher order feature space, the vertices on the interior of the mesh are now located further from exterior vertices. In this manner, interior and exterior vertices will not be collapsed together early in the reduction process. The figure shows the edge lengths that are calculated in the higher dimensional space mapped back to the original mesh. In this example, the weight for each dimension are equal. In a similar manner for meshes used during experimental testing in the following chapter, the initial meshes in $\mathbb{R}^3$ given from the range data are mapped to $\mathbb{R}^{12}$ using the additional multimodal information. The edge lengths are calculated in that space, and the order of the edge collapse determined by then finding the smallest length. This is accomplished by first forming a set of all the calculated edge lengths

$$\mathcal{L} = \{L_{ij} \mid L_{ij} = \vec{w}_{ij}\,|\vec{v}_i - \vec{v}_j| \in \mathbb{R}\} \tag{4.4}$$

The minimum length L_k is found

$$L_k = \min_{ij}(\mathcal{L}) \tag{4.5}$$

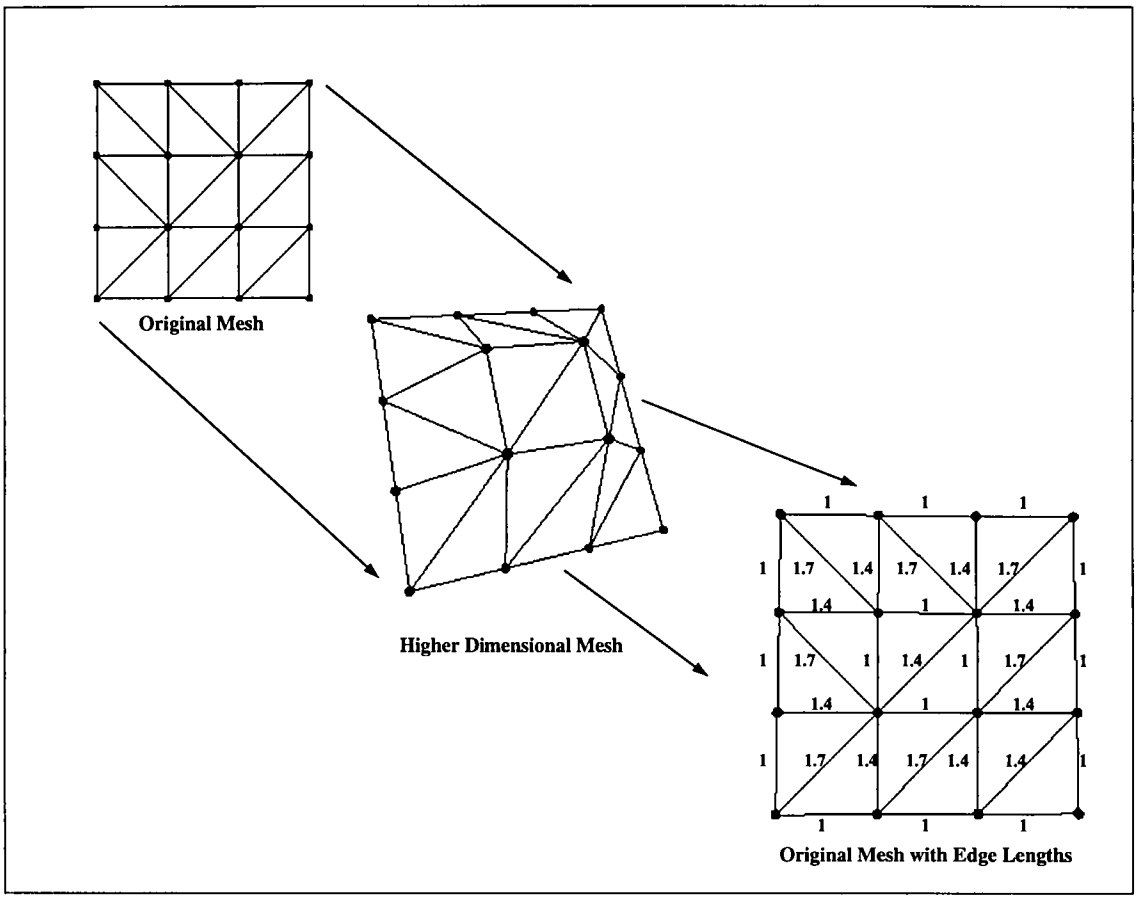


Figure 4.2: *Mapping of vectors into the feature space to calculate edge lengths used in mesh reduction. In this case $\mathbb{R}^2$ is mapped to $\mathbb{R}^3$.*

for iteration k. The edge associated with this minimum length is collapsed and the edge removed from the set

$$\mathcal{L} = \mathcal{L} - L_k \tag{4.6}$$

This is applied iteratively removing the shortest calculated length at each step until the desired reduction is achieved. This feature based approach allows all the multimodal information to be taken into account when performing the mesh reduction.

4.2.2 Data Structures and Representation

A data class is needed for mesh reduction to represent all the features of each vertex. Since the software is written using Open Inventor it would seem logical to use the Open Inventor class structures to represent the data. However, implementation with the Open Inventor classes proved to be very slow due to the large size of the class structure. Therefore, a new simple data class was written.

This project is implemented using C++, and as such, all of the data is stored in classes. The lowest level in the class hierarchy is a *Point*. This structure contains three doubles along with methods to set, retrieve, and manipulate them. It is used for storing geometry, normals, etc. Building on a Point is the *Vertex* class. This is where all the information about the features of the model is stored. This includes the geometric position, normal direction, curvature, texture coordinates, RGB color, boundary angle, and error associated with each vertex in the model. An *Edge* contains the indices of the two vertices which it connects. A *Face* contains the indices of the Edges which comprise it along with a surface normal. Each of these structures is cross-linked to speed data access, so a Vertex knows which Edges are connected to it, and an Edge knows which Faces it is used to form. These structures are then contained in a *Mesh* class which, given the vertices, edges, and faces, knows how to connect them into a mesh. Also, contained in the mesh is the multiresolution data structure which tells the mesh how to move from one resolution to the next. Methods to read various range image formats, including Coleman, FITS, and PGM, and convert them to a mesh are available in the Mesh class. In the process, point and surface normals are generated, along with texture coordinates. Also methods to output Open Inventor models and

multiresolution models are available in the Mesh class. A diagram of how the class structures are organized is shown in Figure *4.3*.

In the data reduction, pattern vectors are used to determine edge length. Therefore, a *Vector* class is available. This is a very simple data structure containing the arbitrary size vectors and the weights associated with them. Another simple *EdgeList* class contains the lengths of the edges used in the reduction. The order of the edges in the reduction is stored in a *Heap* class. A heap is a binary tree data structure designed to quickly sort data so that the smallest data element (or largest depending on the need) is always located at the top node in the tree. Each node in a binary tree has two child nodes. In a heap, the child nodes are always larger than the parent. If a child is smaller than the parent, then the two are swapped. This swapping mechanism allows the tree to quickly re-sort itself when some of the nodes are changed. The Heap class keeps the smallest edge length at the top of the heap and contains methods which automatically update the heap when a node is changed or deleted from the heap, including the use of a heap-sort. This allows the edge lengths to be quickly sorted as they change during the reduction process.

The final data structure is the one for implementing the dendrogram containing the edge collapse and vertex split order. Each node in the class represents a vertex in the mesh. Each level in the dendrogram represents a single edge collapse or vertex split where two vertices are merged into a new vertex. Since we are dealing with faces at the highest level of the visualization, each level in the class contains a list of faces to add and remove. The current resolution of the model is also stored. Forming a linked list of these levels then gives the final dendrogram structure needed to implement the multiresolution model.

Most previous methods use custom built libraries and formats making future expandability and portability difficult. Since we wish to display models on various platforms and also wish to have a standard API, we have chosen Open Inventor as our toolkit. This is a C++ based extension to OpenGL which has hooks to be expanded to included other classes. This library allows for user expansion by allowing the creation of new subclasses of SoNode. As previously mentioned, Open Inventor supports level-of-detail based on multiple models and distance from the view point, but no progressive mesh is available. Therefore, a new Open Inventor node must be created. Here a

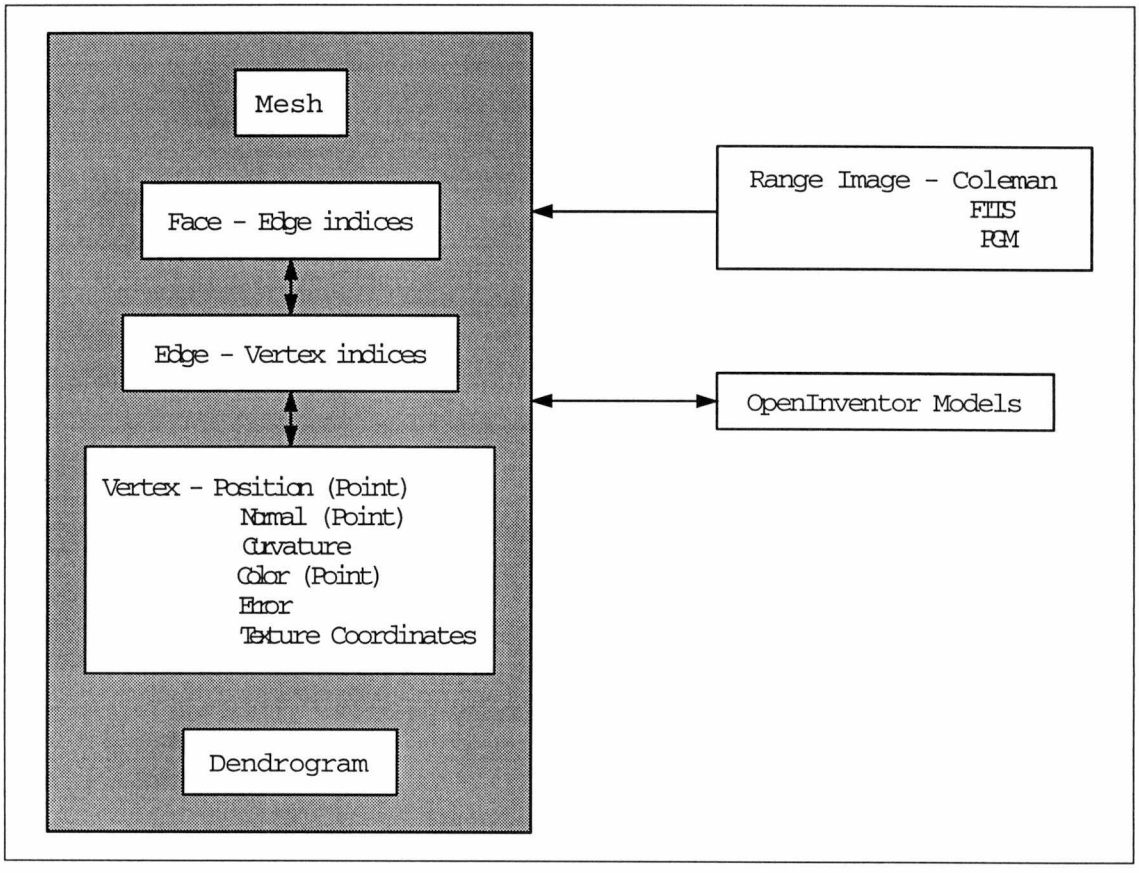


Figure 4.3: *Data structure diagram.*

SoMRFaceSet node is added which contains all the information needed to render a multiresolution node. The Mesh class contains all the information needed to create a multiresolution model. All of this information is obtained off-line. The SoMRFaceSet is a subclass of the SoShape node. It contains several fields, including those for minimum, maximum and current resolution of the model contained within the node, texture coordinates, normals, materials, vertices, face sets, desired frame rate, and the information for the dendrogram previously described. One of the methods for the Mesh class is a function which writes a SoMRFaceSet from the mesh data. One of the goals of this research was to have a constant interactivity level for the user which is measured as frame rate. To accomplish this, the resolution of the model is controlled automatically. Details of creating an Open Inventor node can be found in [115].

4.3 Reduction Implementation

The reduction method is started by creating the feature space, or pattern space, $\vec{v}$from the vertices V. This space currently contains geometric position, point surface normals, point curvature, color, and boundary information. Each of these features has an associated weight which the user sets based upon which feature is deemed to be most important for the particular model. The edge lengths L_{ij} of the initial mesh are then calculated and stored in the heap. With this step completed, the iterative edge contraction begins. At this point in the reduction process, several steps occur simultaneously. These include the edge contraction, calculation of the error, and creation of the multiresolution model. The edge on top of the heap is to be contracted into a vertex and is marked for deletion. To preserve boundaries, several cases are considered. First, if the edge is a boundary edge, both of the vertices connected by the edge are also boundary, or external, vertices. In this case, we wish to preserve the sharpest point, the one which has the greatest angle formed by the two boundary edges connected to it. Therefore, in this case, the edge is collapsed to the vertex with the greatest angle. The second case is where one vertex is external and one is internal. Again, here we wish to preserve the boundary, so the edge is collapsed to the external vertex. The last case is where both vertices are internal. Here, the edge is collapsed to a new vertex which is in the middle of the edge and a new vector formed in the feature space to reflect this addition.

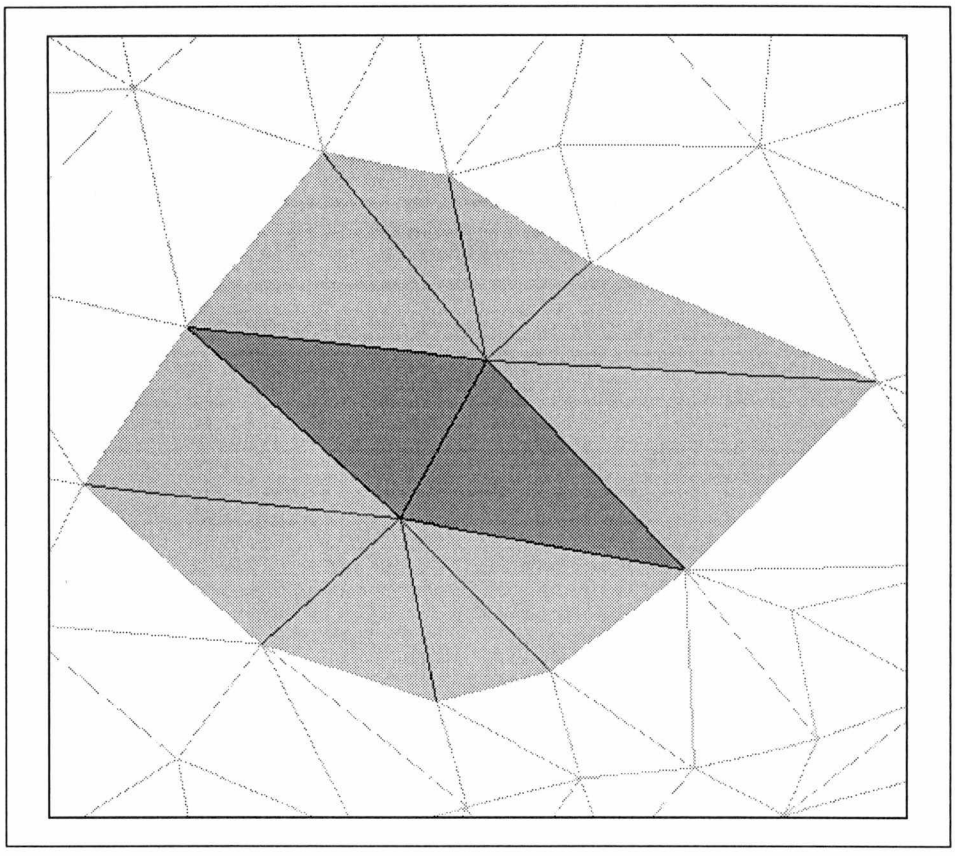


Figure 4.4: *Edges and faces marked for removal (darkly colored) and update (lightly colored).*

Figure *4.4* gives an example as to how the edges and faces are marked for update and removal. The edge in the center of the image is marked for collapse. When an edge is collapsed, one or two faces are removed. At the same time duplicate edges can be formed. This occurs because the remaining edges which helped form the removed faces now have the same end points. Therefore, one of these edges must be removed and is marked for deletion, and the other is updated with neighbor information of the first. So the dark faces in the figure are marked for removal and the lightly colored faces are marked for update. Likewise, the lightly colored edges are also marked for update. The dark edges are marked as duplicates, so one is removed and one updated. At this point, all edges which have been marked for deletion are removed from the heap, and the heap updated accordingly. Also, the vertex edge lists are updated. All of the neighboring edges are marked for update since the length of these edges have changed with the vertices moving and the

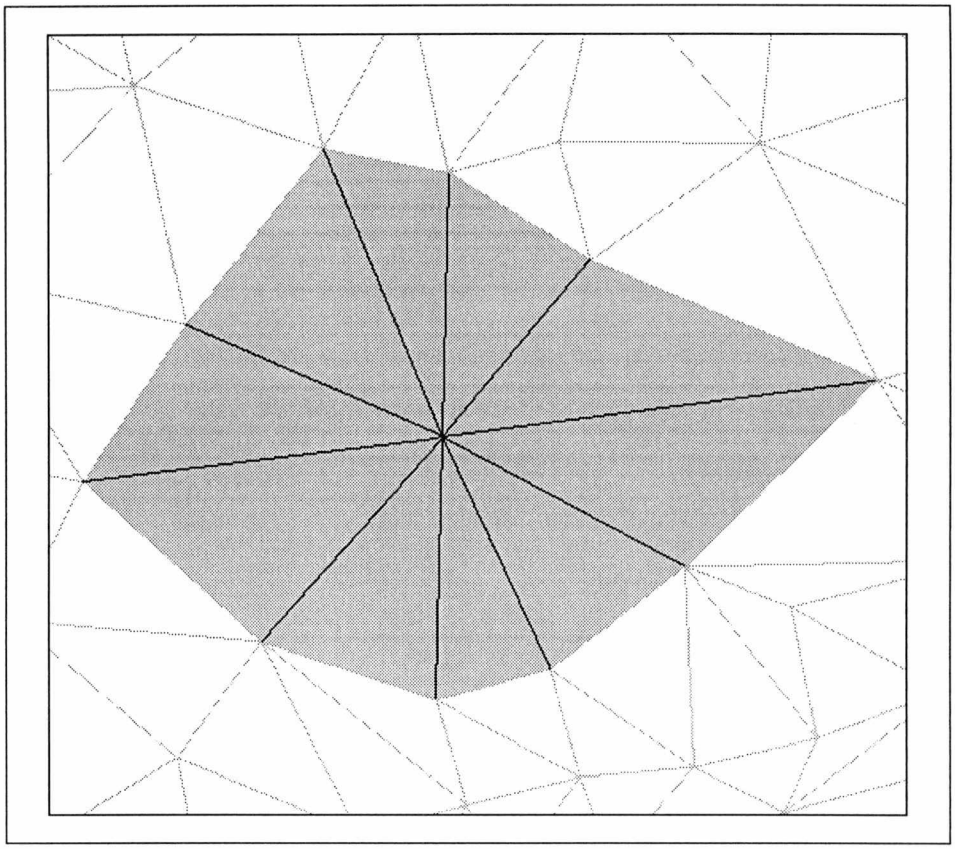


Figure 4.5: *Resulting edges and faces after removal and update.*

heap updated automatically moving another edge to the top of the heap.

Figure *4.5* shows the results of the updated mesh. The next step involves the creation of new edges and faces for the multiresolution model. This is done during the reduction process instead of at run time so no computational time is needed to update the model while being viewed. All of the edges which where marked in the last step are used to create the new edges, and the faces which they comprise are created as well. Surface normals for these new faces are calculated. At this point the error from the current iteration can be calculated as described in Section *4.3.1* and the maximum error updated. Also, at this point, the new faces being created can flip over neighboring faces. By comparing the normals of the new face to the old face, an illegal face flip can be detected. If one is detected, the algorithm reverts back to the previous state before removing the last edge, adjusts that edge's length and thus the heap, and proceeds with the collapse using

the new shortest edge. Once this is complete, the edge lengths of the marked edges are updated and the heap adjusted accordingly.

At the end of each iteration, a level of the multiresolution representation is formed. This includes setting a resolution, level number, and error in the multiresolution representation. Also included is a list of faces removed and new faces created. This iteration process is continued until no more reduction can be performed or a preset threshold is reached. The threshold can be based on error, reduction ratio or percentage, or minimum number of triangles. At this point the reduction is complete and the resulting multiresolution model has been formed. A flowchart of the overall method is given in Figure *4.6*.

4.3.1 Error Measurement

Many different error measurements are used throughout the literature to determine the quality of the models produced by polygon reduction. The simplification envelope method [18] itself is based on a percentage error since the offset surfaces used in creating a reduced model are actually error bounds. Most other methods, however, try to keep track of some type of maximum or worst case error using a Hausdorff estimate or vertex displacement measure. We have chosen a vertex displacement measure based on the distance from vertices removed during an edge collapse to new faces created at each step of the reduction. This will measure the maximum possible error, the actual error may be less. The error measurement does not enter into the calculations for the mesh reduction but is merely a means by which to determine the quality of the model at a given point in the multiresolution hierarchy. We determine the distance from the old vertex to the new face by dropping the vertex parallel to the normal of the face. This distance is determined using the inner product:

$$D_n = \left| \frac{(\vec{V}_n - \vec{V}_o) \cdot \vec{F}_n}{\|\vec{F}_n\|} \right| \tag{4.7}$$

where $\vec{V}_n$ is a vertex on the new face, $\vec{V}_o$ is the old vertex, and $\vec{F}_n$ is the normal of the new face as shown in Figure *4.7*. In our case, the normals are unit length, so the distance can be calculated using:

$$D_n = \left| (\vec{V}_n - \vec{V}_o) \cdot \vec{F}_n \right| \tag{4.8}$$

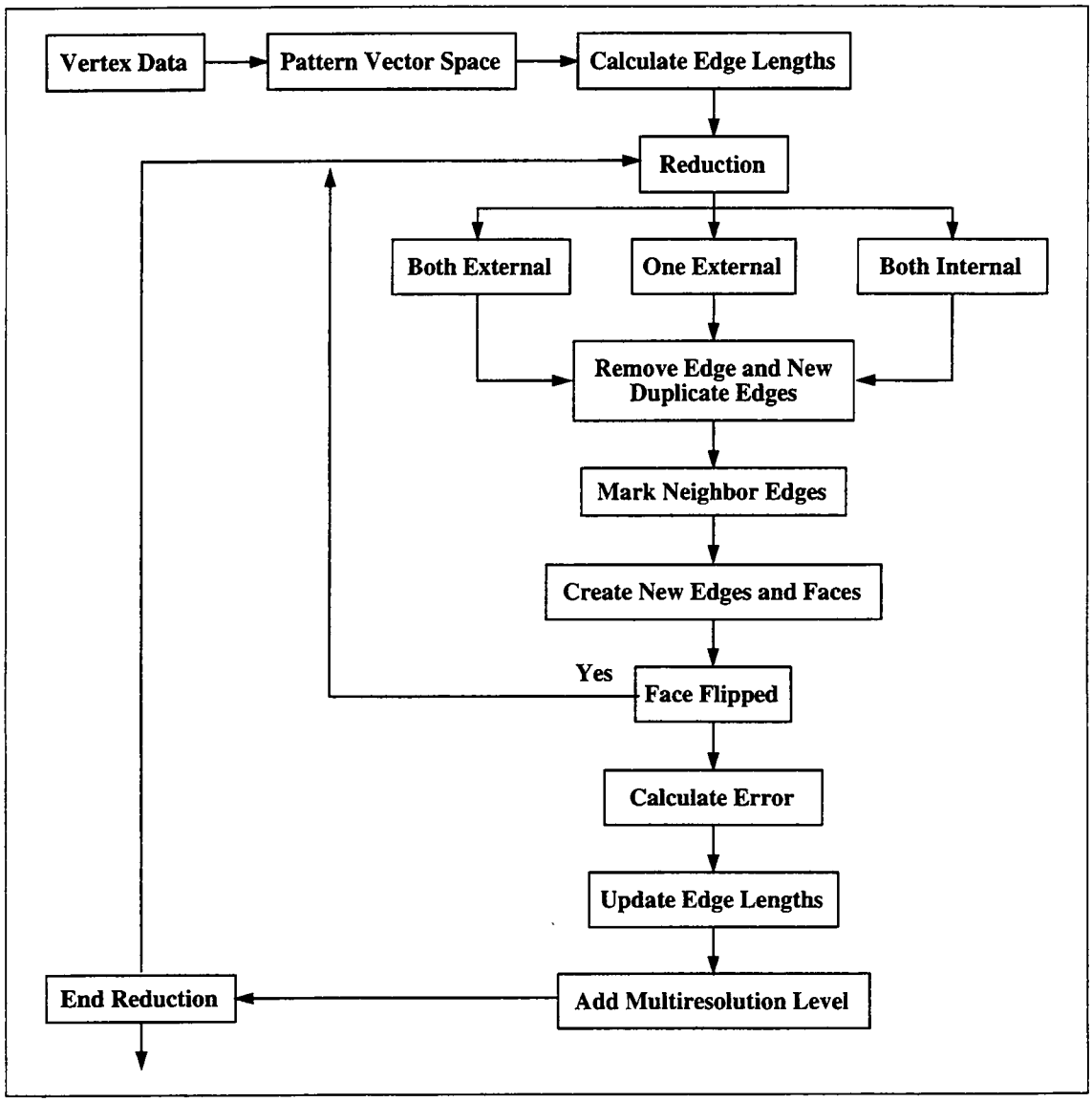


Figure 4.6: *Flowchart of the reduction method.*

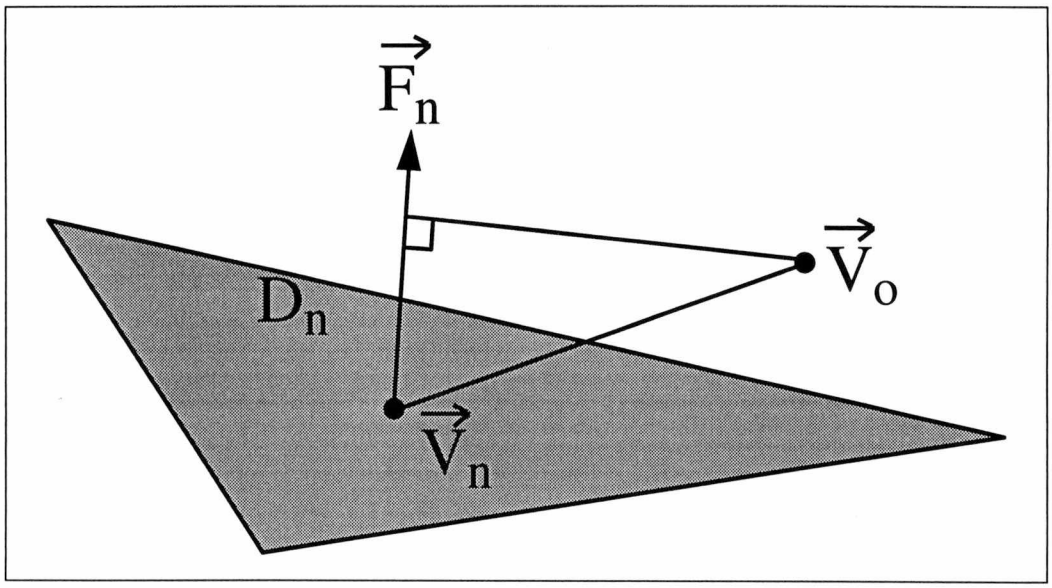


Figure 4.7: *Calculation of error created by removing faces.*

At each step in the reduction method the distance is calculated from the vertices connected by the collapsed edge, V_i and V_j, to the new faces created by collapsing the edge, $f_k ... f_{k+l}$. The minimum distance or error E_i at a vertex V_i is calculated as

$$E_i = \min_k (D_k) \tag{4.9}$$

This error is added in with any previous error already associated with the vertex. The error E_k at the new vertex created V_k is assigned the maximum of the errors E_i and E_j. In this manner, error can be accumulated by adding any additional error incurred by collapsing an edge containing V_k. Therefore, when V_k is collapsed into V_m,

$$E_k = E_k + \min_m (D_m) \tag{4.10}$$

This allows an accumulated estimate of the maximum error at each step in the reduction. In addition, we also want to employ a percentage error of the model to be used as a stopping criteria in the reduction process. This presents a problem because the error needs to be a percentage of some measurement. We have chosen to represent the percent error with respect to the distance across the diagonal of the bounding box of the original model. Since the unit of measurement in a model can be arbitrary, this gives a common ground with which to measure the relative error

between different models. Figure *4.8* show an example of a model with its associated bounding box. In this particular case, the diagonal across the bounding box measures 89.988 units.

This geometric error measurements give an idea of how much error is being incurred in the reduction, but for visualization what is really needed is more of a visual measurement. We need to know if the reduced model "looks" like the original model. There is really no good way to determine this except by subjective means. One method is to project the models into an image space and then use a metric on the difference of the two projections to produce an overall quantification of the subjective quality of the reduced model. This will give a measure on a per pixel basis. Figure *4.9* gives an example of this applied to a simplified model created from a synthetic range scan. This shows the original model and a model reduced by a factor of 10. The difference between the two projected images show visual errors along the boundary of the model. The most notable is the right front wing of the plane. By thresholding the subtracted image, the areas where larger error is occurring is viewed more easily as shown in Figure *4.10*.

4.4 Conclusions

Once again, our ultimate goal is the real-time display of large multimodal laser range data sets for multiple users. The multi-user aspect was accomplished through the design and implementation of the MERLIN visualization system. This leaves the real-time display portion of the goal. To accomplish this requires the abilities to create multiresolution models from our data, which, in turn, requires a mesh reduction algorithm. Our requirements for this algorithm were that it be a decimation method capable of handling multimodal data, able to output a multiresolution model using a common data format, and be easily extendable for future data modalities. To fulfill these this requirements, this chapter has described the implementation of a new pattern vector based triangle mesh reduction algorithm, POLYMUR, which can take into account all of the data parameters with which we are dealing. Using the initial vertices, an n-dimensional pattern space is derived which accounts for all the data modalities. The edge lengths for the initial edge connections are calculated in the new feature space. With these lengths, edge contraction is then performed starting with the shortest edge and iterating to a terminus. The output of the algorithm

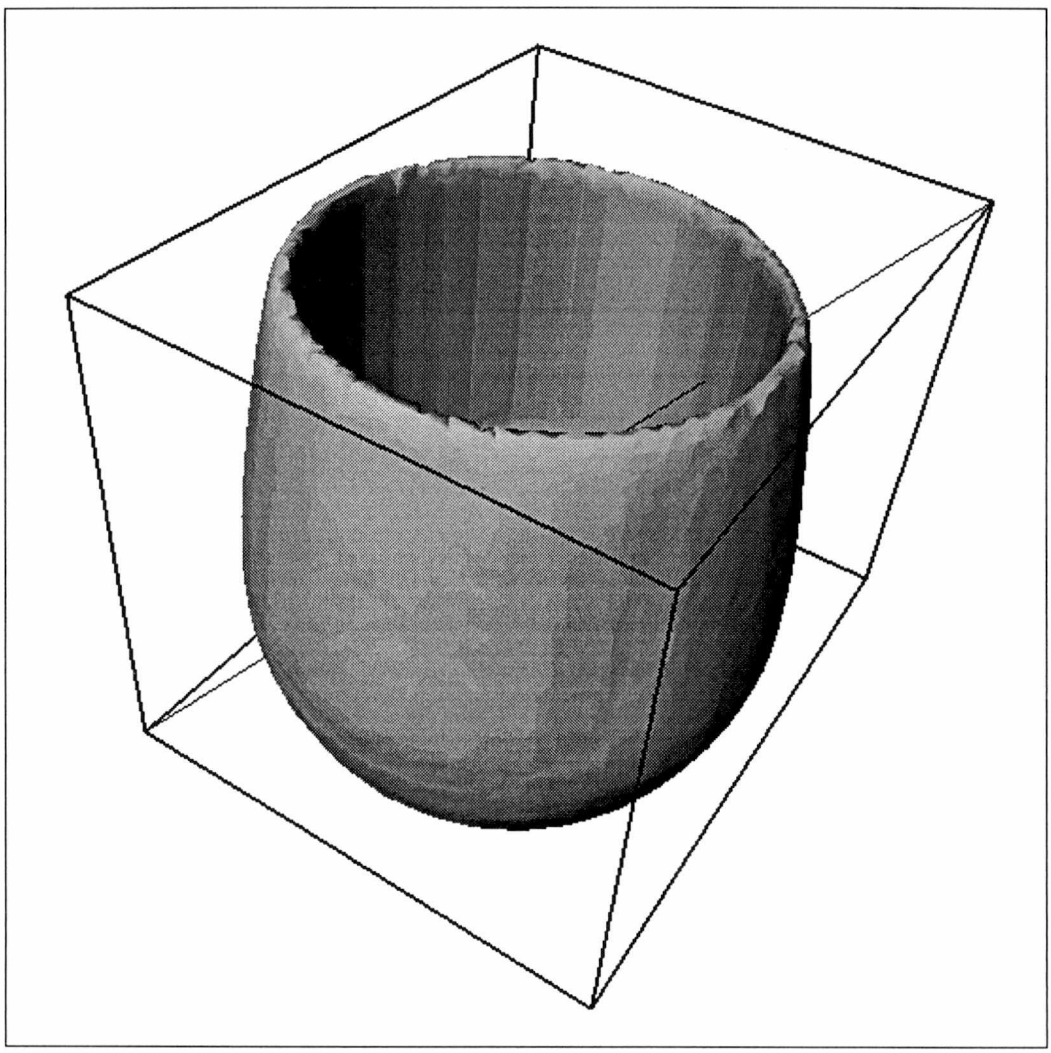


Figure 4.8: *Model shown with its associated bounding box used in percentage error calculation.*

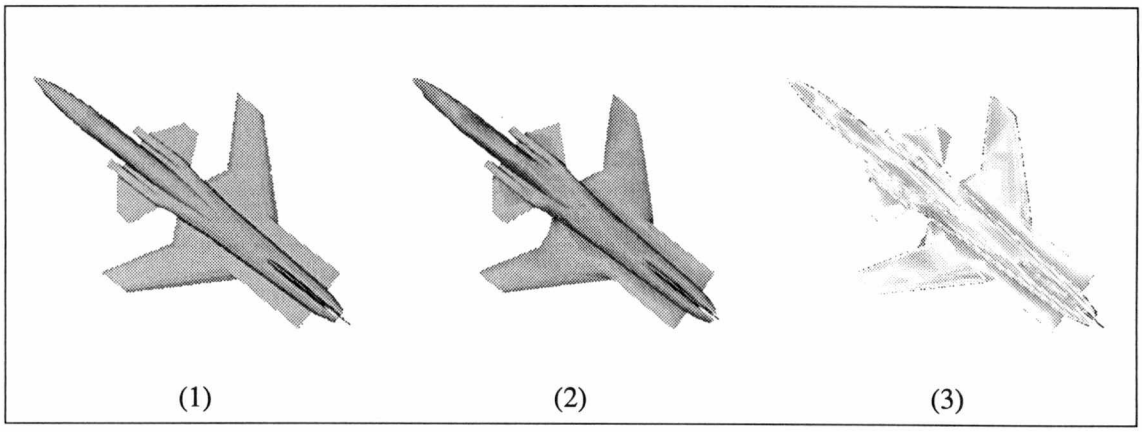


Figure 4.9: *(1) Original model created from a synthetic range scan, (2) model after 90.2% reduction, and (3) the difference from the two projective views.*

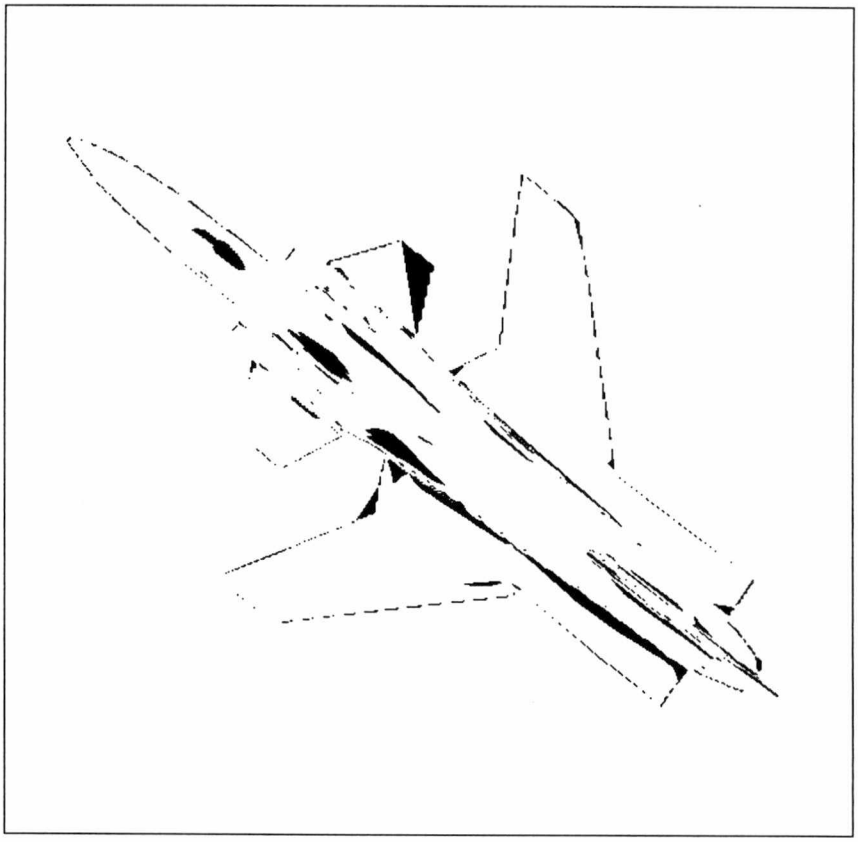


Figure 4.10: *Thresholded version of the visual error image showing the areas of change between the original and reduced models.*

is in a standard data format, Open Inventor. Automatic resolution selection of the multiresolution model is then used to maintain a pre-selected frame-rate when interacting with the model. Compared to other mesh reduction algorithms, this method has several advantages. First is the ability to handle multimodal data. A few of the reviewed techniques handle more than just geometrical information, but none can be extended easily to include any new data modalities. Also, the pattern based method can be easily tuned by the user to emphasize any feature of the data by simply adjusting the pattern weights. Most techniques also do not output multiresolution, or continuous level-of-detail, models. In addition, the pattern based reduction is capable of handling a variety of models instead of being restricted to manifold objects or height fields as some methods are. The final advantage of this method is that the output is in a common data format. This allows anyone to use the multiresolution model once it is created. The next chapter gives many examples and results of the algorithms using a wide variety of models.

CHAPTER 5

Experimental Results

As stated previously in Chapter *2*, the display rate is the main limitation for the low-cost CAVE system. The proposed fix for this problem is the implementation of a novel, pattern vector based mesh reduction algorithm which is multiresolution capable as presented in Chapter *4*. This chapter gives the results of the most substantial part of the research presented in this dissertation, the pattern vector based mesh reduction algorithm. Included is a section giving the results of calculating the initial edge lengths based on the varying weights that are possible for the pattern vectors. This is followed by several examples of the reduction on a variety of data sets. Finally, the fixed-rate interactivity is presented and discussed.

5.1 Feature Based Edge Length Calculation

Recalling that the reduction order of the edges is determined by the distance between the end points of the edges in the feature space, the weights chosen for the vectors directly effect these distances and thus can cause the reduction to emphasize one feature over another in the model being reduced. Figure *5.1* shows how each of the weights effect the edge lengths in a model. Here a model created from a single synthetic range image is used. The initial calculated edge lengths are color coded according by size with red being the shortest length and blue the longest. Therefore, in each case, the edges which are colored red will be the first to be collapsed. For example, the model using only boundary information has all internal edges colored red and all external edges blue, where the model using only curvature has flat surfaces colored red and fades to blue as the curvature increases. It should be noted that before the reduction iterations begin, the pattern vectors are normalized. The reason for this is that the unit of measurement for the geometric position of the vertices is completely arbitrary, some models are measured in meters, some in inches, some in kilometers. Therefore, the vectors must be normalized so a distance

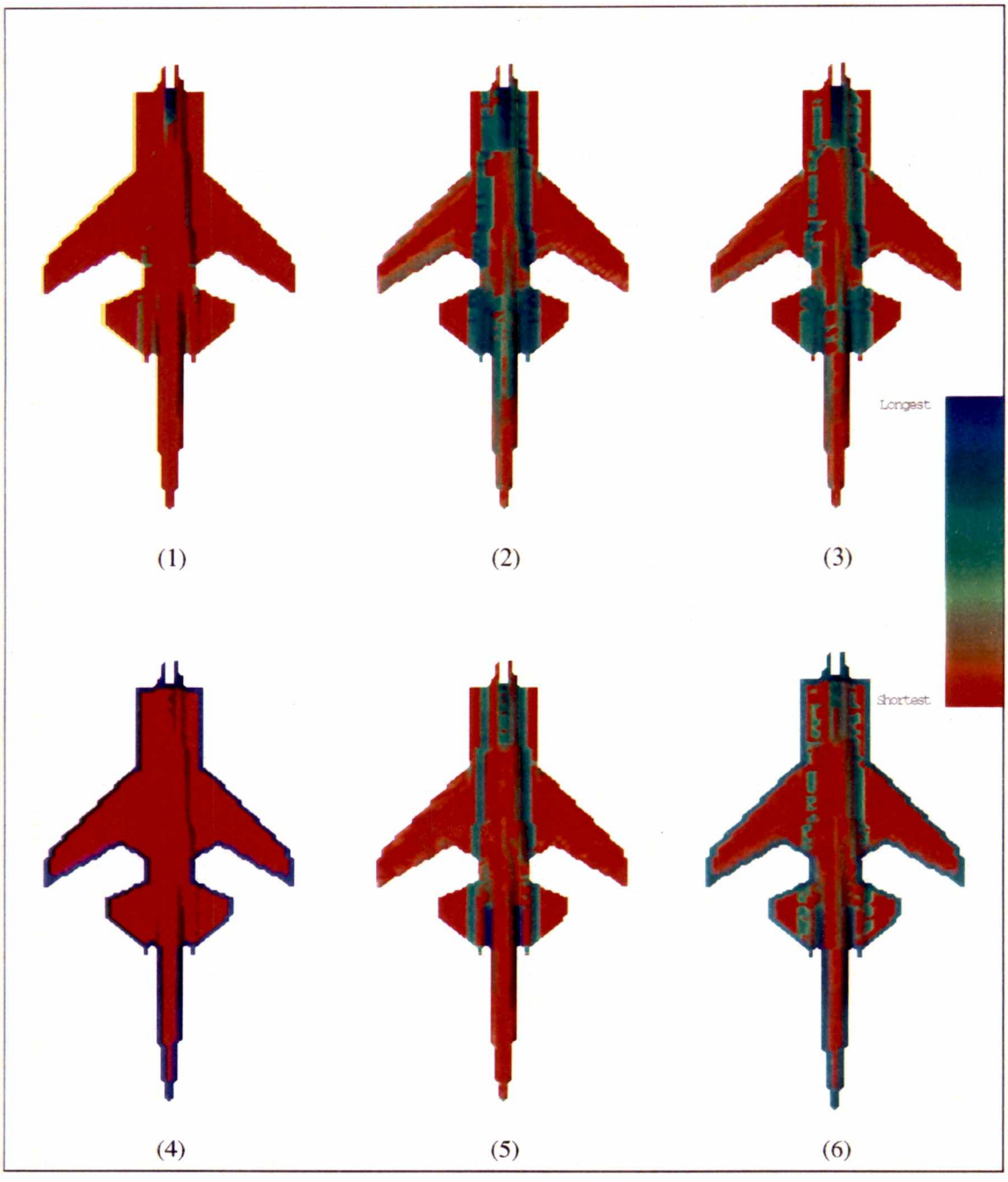


Figure 5.1: *Color coded edges lengths based on weights using (1) geometry only, (2) normal only, (3) geometry and normal, (4) boundary only, (5) curvature only, and (6) all vectors weighted equally.*

of 10,000 millimeters does not cause the influence of features such as normal and color which are already normalized to be unsubstantial. Figure *5.2* gives an example of weights that would typically be used in model reduction. In most cases we wish to keep areas which are feature rich. These are areas which have high curvature or boundary edges. Therefore, typical weights would have higher values for curvature and boundary information. However, in some cases, as will be shown in Section *5.2.2*, other information may take precedence. There is no optimal set of weights which works for all models in all situations. The weights, however, are very intuitive in what features they effect. Therefore, the choose of weights for a specific model only needs to be determined by which features are to be preserved. It should be stated that the edge lengths shown in the figure are the initial calculated lengths. As the algorithm progresses and vertices move, the edge lengths are updated based on the new end points created.

5.2 Pattern Vector Based Mesh Reduction Implementation Results

The new reduction algorithm is tested on a variety of models created from several different methods. A few of the data sets tested are presented in this chapter to give an idea of the resulting reduced models. Models created from individual range images, both synthetic and real, and from a digital elevation maps are tested. Also, models created from fused data sets of multiple range image views are used for testing. First, a synthetic data set is presented. Then a real range data set from the Perceptron laser range camera and a registered segmentation image is given followed by a Perceptron range image and a registered color intensity image. Also, a range data image taken with a Coleman laser range scanner is presented followed by a digital elevation map model. Finally, the last data set is a full 3D model created from multiple range data images. This gives a wide variety of models to test the robustness of the algorithm. It should be noted that this reduction method begins to break down as far as preserving the visual integrity of the model as the number of polygons approaches zero. This is evident in all the models tested when the reduction is over 99%. This is also true of all reduction methods and is to be expected since a finite number of polygons are needed to represent a real model.

Also included is timing information of the various models. The reduction algorithm was run

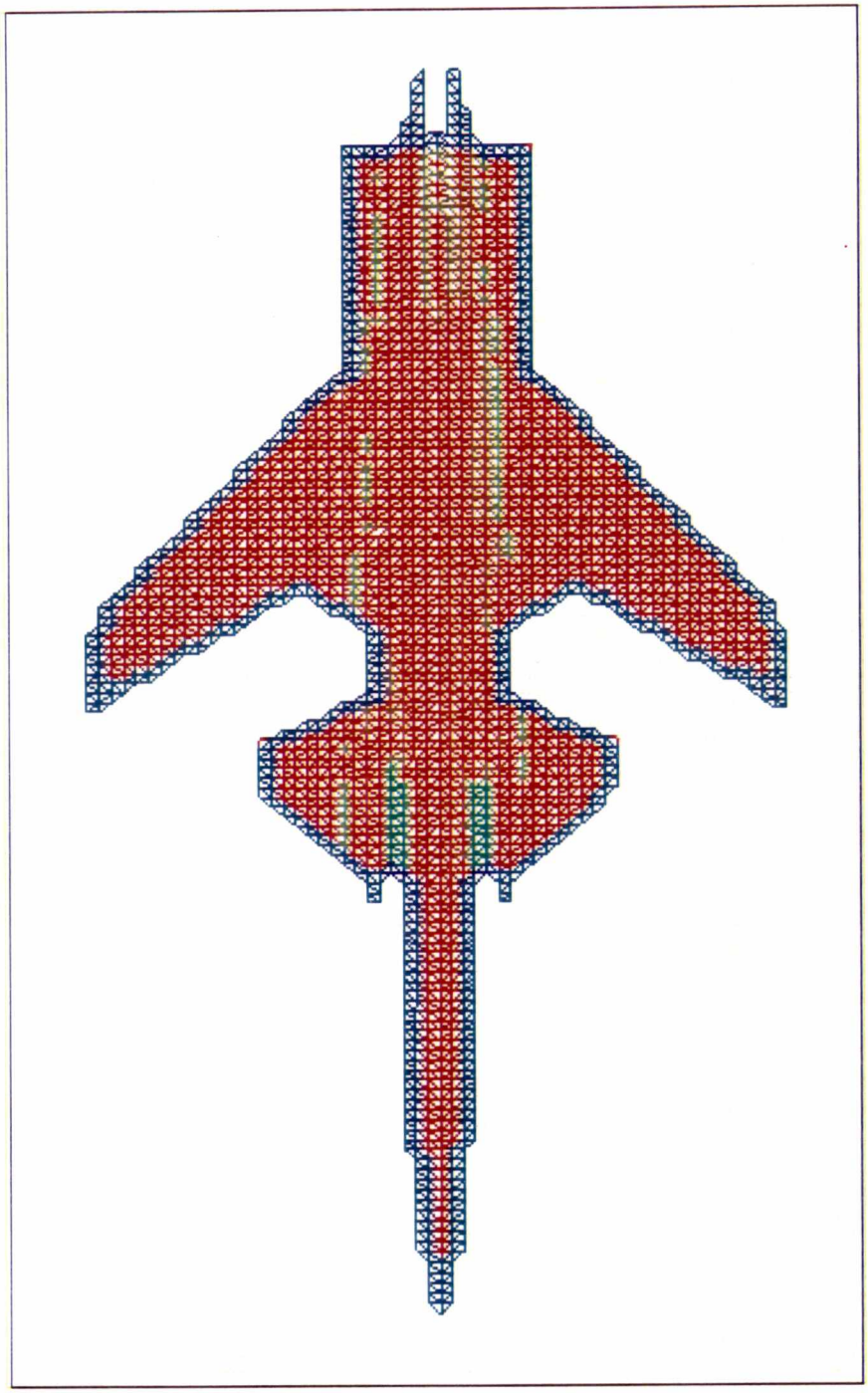


Figure 5.2: *Typical vector weighting with the geometry weight = 1, normal weight = 0.5, boundary weight = 2.0, and curvature weight = 1.5.*

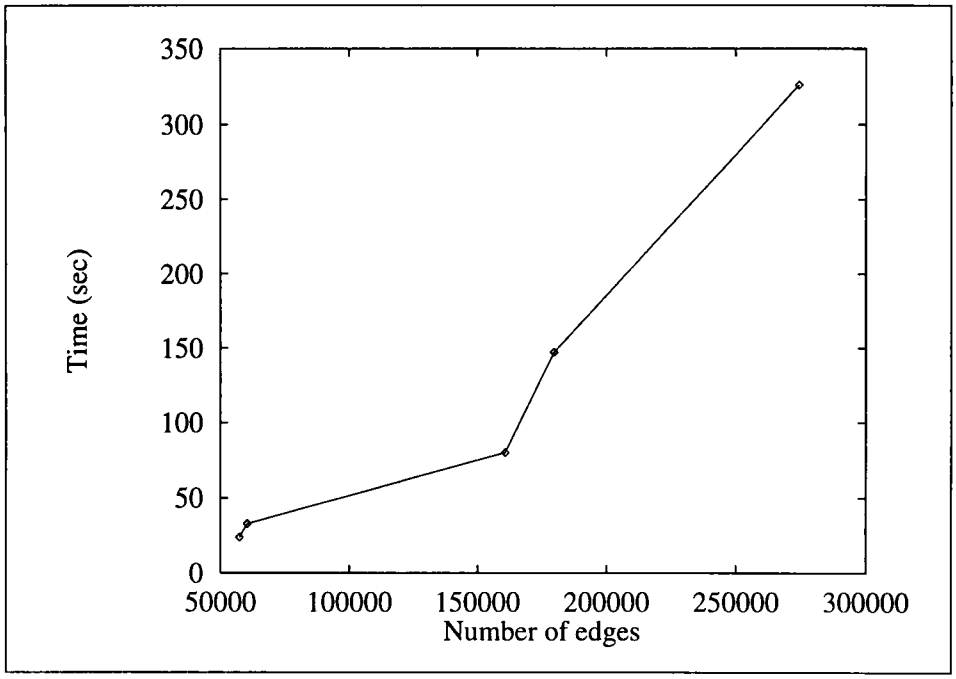


Figure 5.3: *Amount of time versus the number of edges collapsed for the various models.*

on two different machines. The first is a Silicon Graphics Indigo2 MaxImpact with a 195MHz R10000 processor and 128MB of RAM. The second is a Silicon Graphics Onyx with two 194MHz R10000 processors and 512MB of RAM with two-way interleave (effectively 256MB of RAM). The two machines show the same reduction time for the smaller models. The Onyx, however, is much faster on the larger models because of the faster disk access when memory swapping begins. Figure *5.3* shows the time that the reduction algorithm takes to complete. There are two distinct linear regions on the graph. The first is where swapping is not needed and the second is where the data is swapped from memory to disk. Because of the disk swapping, another item of interest is the amount of memory used in the reduction since we are dealing with large data sets. For the models tested, between 1.8KB and 2KB of memory is needed for each edge in the initial model if reduction is continued to 100%. This is due to the number of new vertices, edges, and faces that are created during the reduction process and the overhead associated with the algorithm. For example, the range image shown in Figure *5.4* results in a mesh containing 160,698 edges, 104,411 faces, and 65,536 vertices. This mesh requires a maximum of 305MB of memory while



Figure 5.4: *256x256 range image resulting in a model containing 160,698 edges and requiring 305MB of memory for reduction.*

running for complete reduction of the model.

5.2.1 Synthetic Data Model Results

The model presented here is created from a synthetic range scan. It should be noted that in this case no noise is added to the range image. This image is merely used as a first test of the reduction algorithm before real data is tested. This model is created from one range image, so boundary information is important. The pattern vector reduction was run using a weight of 5.0 for the geometry, 0.5 for the normals, 6.0 for the boundaries, and 3.0 for the curvature. In this case, color has no effect since each vertex has the same color.

The results of the reduction are shown in Figures *5.5*, *5.6*, and *5.7*. Both solid and wire-frame models are shown so the underlying structure of the model is visible. The reduction took about 1 minute and 10 seconds running on a Silicon Graphics Indigo2 with a 195MHz R10,000 processor and 128MB of RAM and the same amount of time to run on a Silicon Graphics Onyx2 with two 194MHz R10,000 processors and 512MB of RAM. As can be seen in the figures, the weights chosen allow the flat surfaces of the model to be reduced first. With over half of the initial triangles

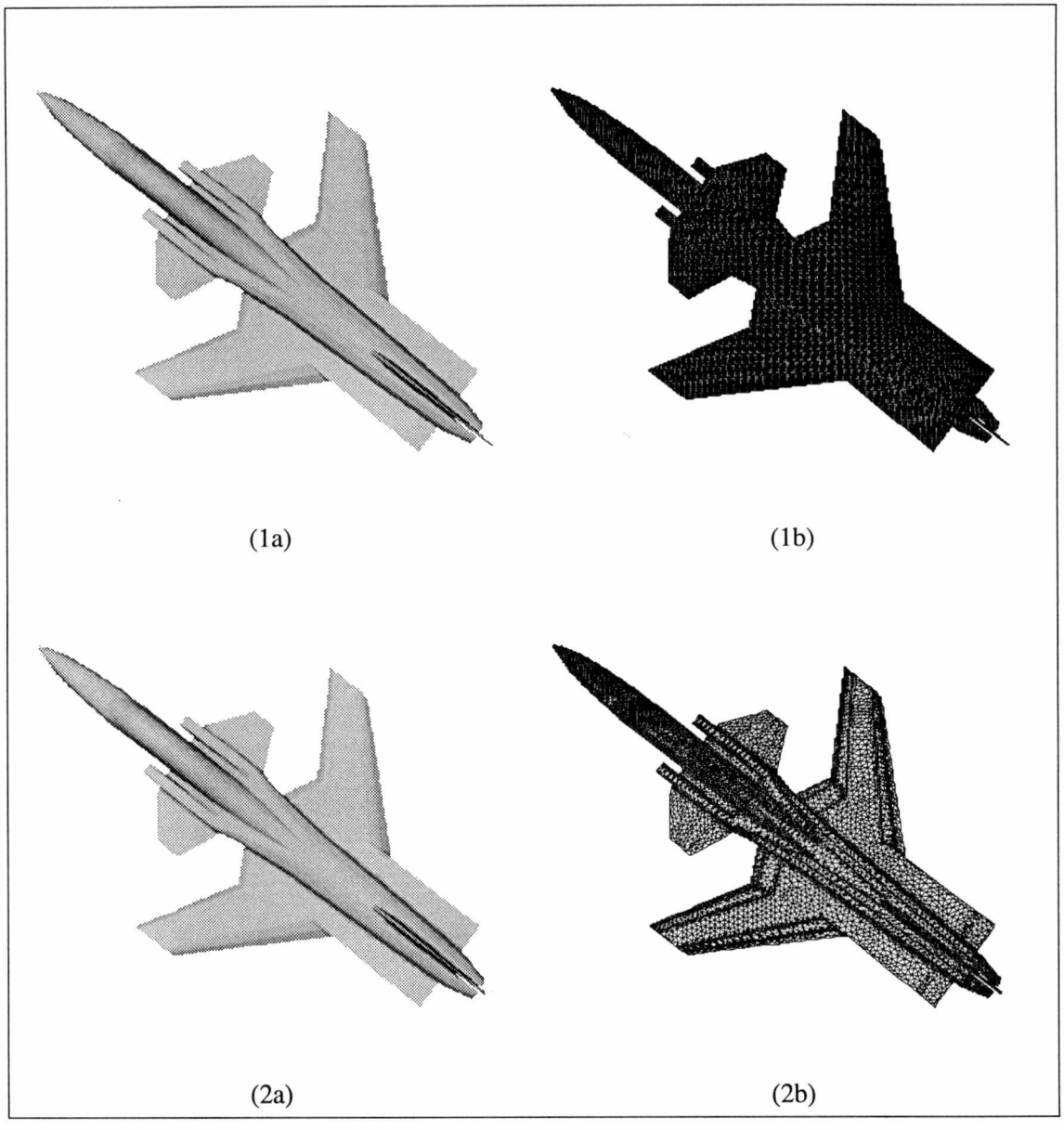


Figure 5.5: *(1a) Initial model from synthetic range data with 39,966 faces, (1b) wire-frame of the initial model, (2a) model after 60.3% reduction, and (2b) wire-frame of the 60.3% reduced model.*

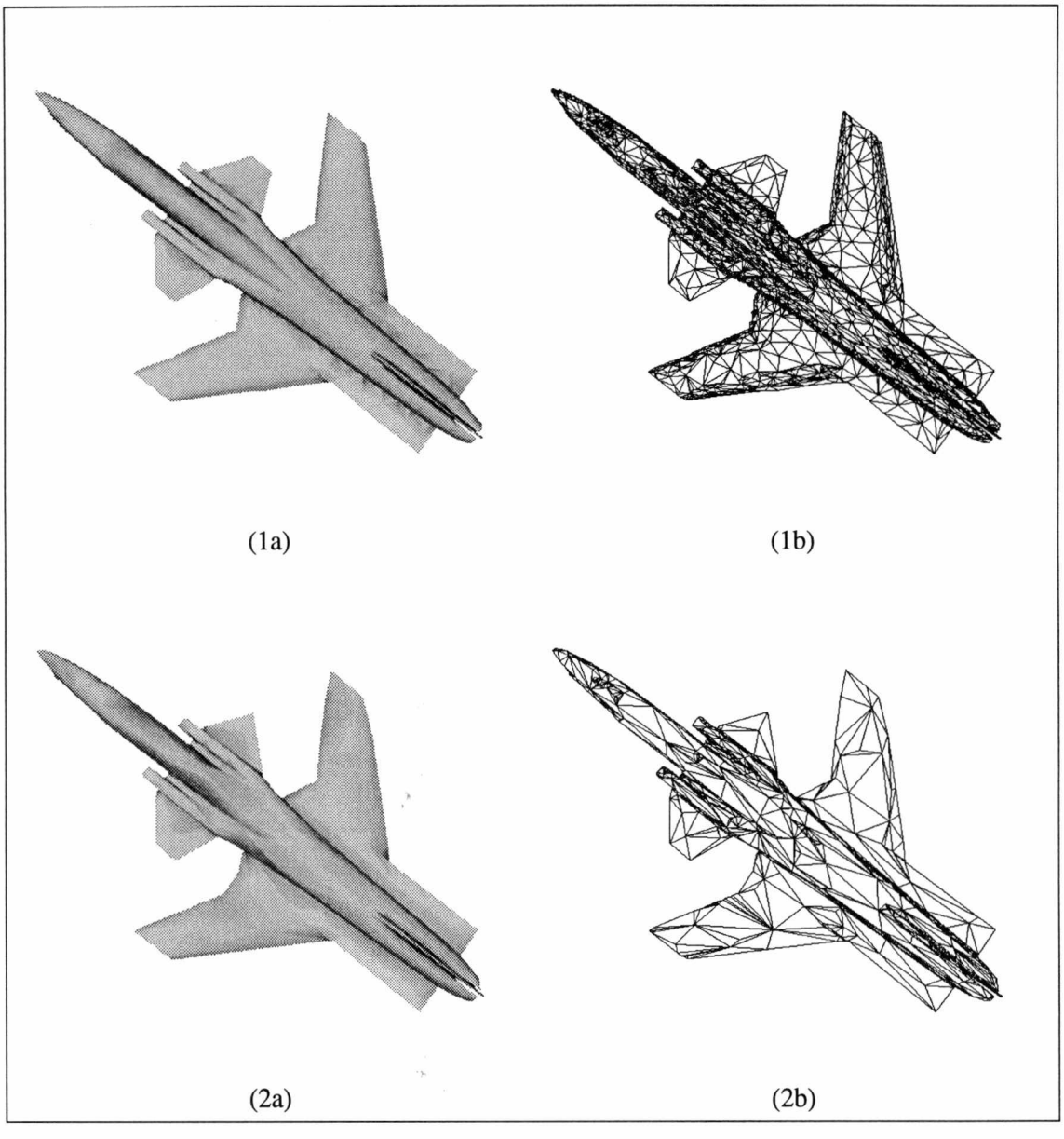


Figure 5.6: *(1a) Model after 90.2% reduction, (1b) wire-frame of the 90.2% reduced model, (2a) model after 97.8% reduction, and (2b) wire-frame of the 97.8% reduced model.*

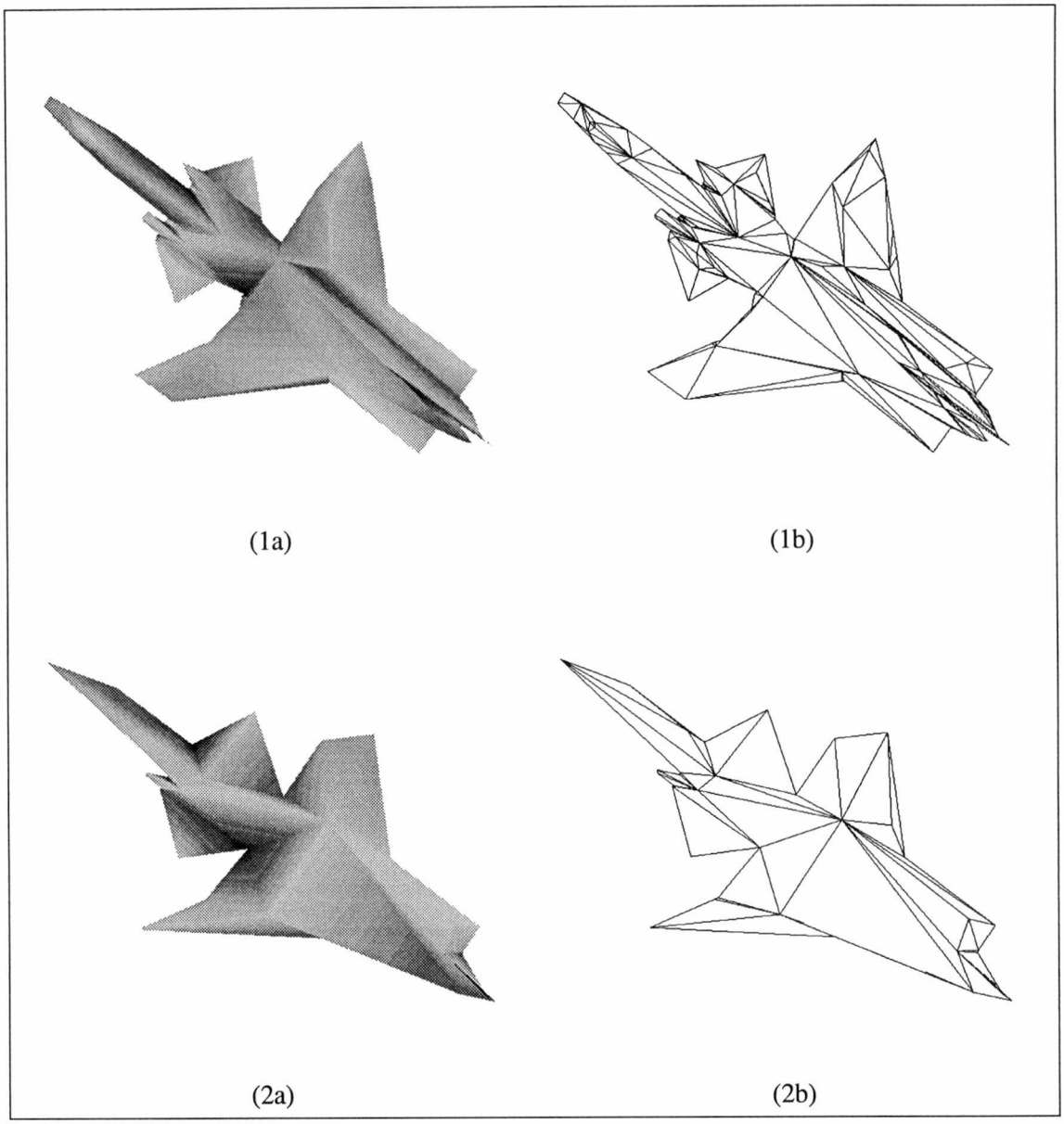


Figure 5.7: *(1a) Model after 99.6% reduction, (1b) wire-frame of the 99.6% reduced model, (2a) model after 99.9% reduction, (2b) wire-frame of the 99.9% reduced model.*

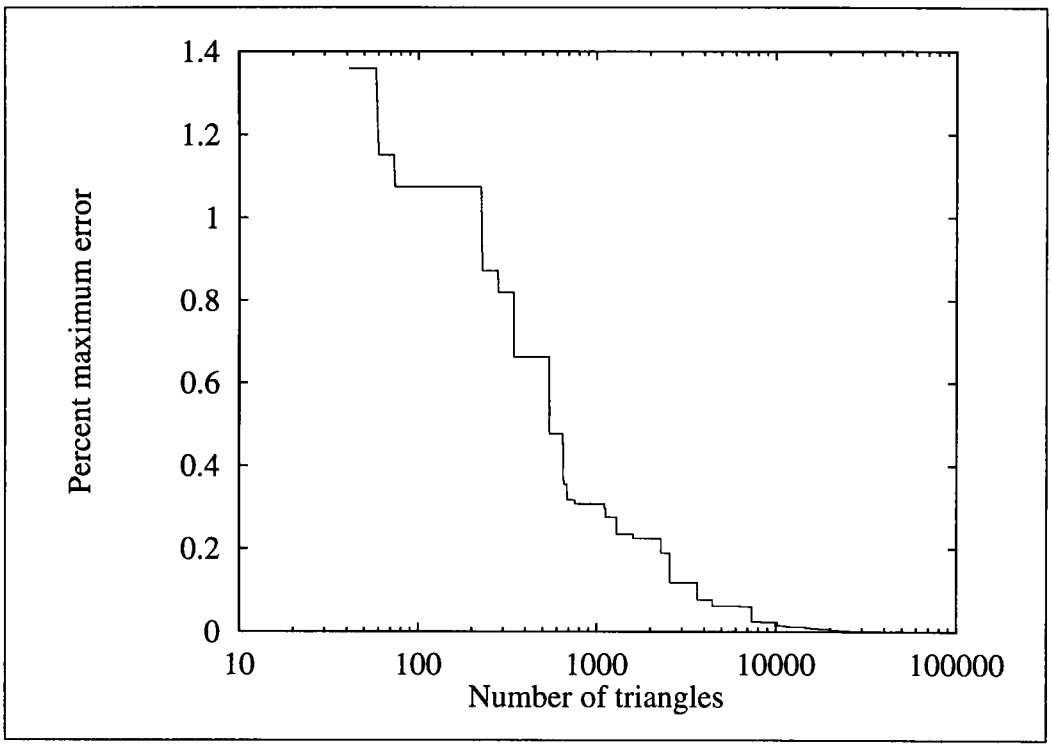


Figure 5.8: *Maximum calculated error versus the number of triangles for the model created from a synthetic range image.*

removed (60.3% reduction), the model still appears almost exactly the same. At around 90% reduction a few differences from the initial model begin to appear. With more than 99% reduction, the model is no longer perceptible as an airplane. The error incurred during the reduction as defined in Section *4.3.1* is shown in Figure *5.8*. As can be seen, the error is almost exponential with respect to the number of faces in the model as the reduction proceeds to completion.

5.2.2 Perceptron Range Data with Segmentation Information

The data set presented here is created from an image taken from the Perceptron laser range scanner (see Figure *5.9*). The Perceptron returns both a range and a registered intensity image of the scene. From the range image, a triangle mesh is created as described in Appendix *A.6*. The laser data is rather noisy. To compensate for this, the range image is median filtered before the 3D mesh is created. The resulting mesh contains 116,544 faces and 179,536 edges. The mesh, however, does contain holes where large amounts of noise could not be effectively removed. This is because the

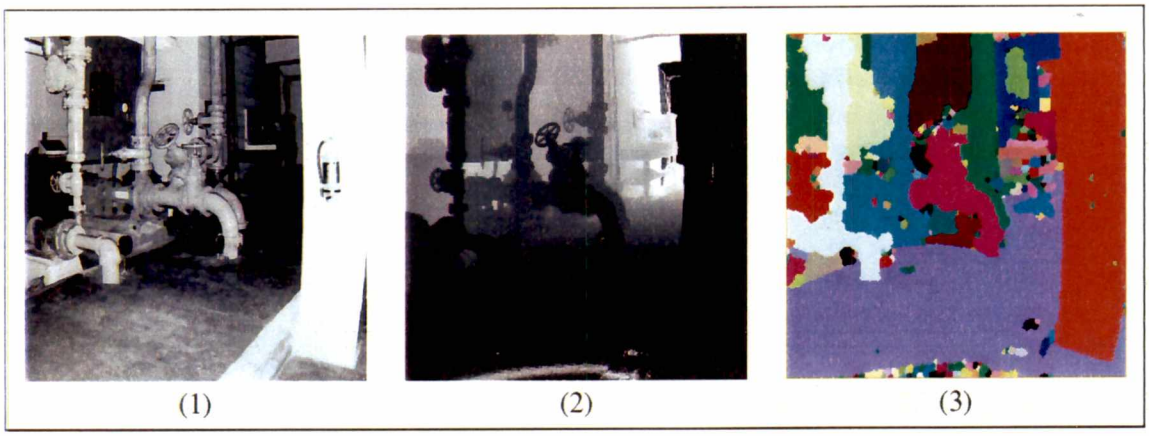


Figure 5.9: *(1) 256x256 range image taken with the Perceptron laser range camera only, (2) registered intensity image taken with the Perceptron laser range camera, and (3) segmentation of the range image.*

mesh creation algorithm does not connect the noise point when the resulting triangle would be longer then the preset threshold, thus leaving a hole. These holes are one reason for the need to preserve boundaries in the meshes with which we are dealing.

Parallel efforts at the IRIS lab are focused on range image segmentation. An example of a segmented image by Burgiss [11] is given in Figure *5.9*. The segmentation is incorporated into the reduction by using the segmented image for the color information vector. By setting the color vector weight extremely high, reduction across segmentation boundaries will not occur.

The results of the reduction are given in Figures *5.10* and *5.11*. The mesh is texture mapped with the registered intensity image. Errors at the bottom and right side of the range image show up as small spikes in the model. These can be filtered out during the mesh creation, but this also removes non-noisy data. Therefore, the small spikes are left in the data. Large spikes, however, are removed. A zoomed view of the scene is given in Figure *5.12*. This gives a good example of exactly how noisy the models created from real data are and is a very good example of the types of problems encounter when reducing real data. As can be seen by the reduced models, the removal of the large spikes results in holes in the model. Also, the numerous objects in this scene add more holes to the model from shadows. To compensate for this, the value of the boundary weight is increased. Since the models are not really smooth, the normal and curvature values are lowered to

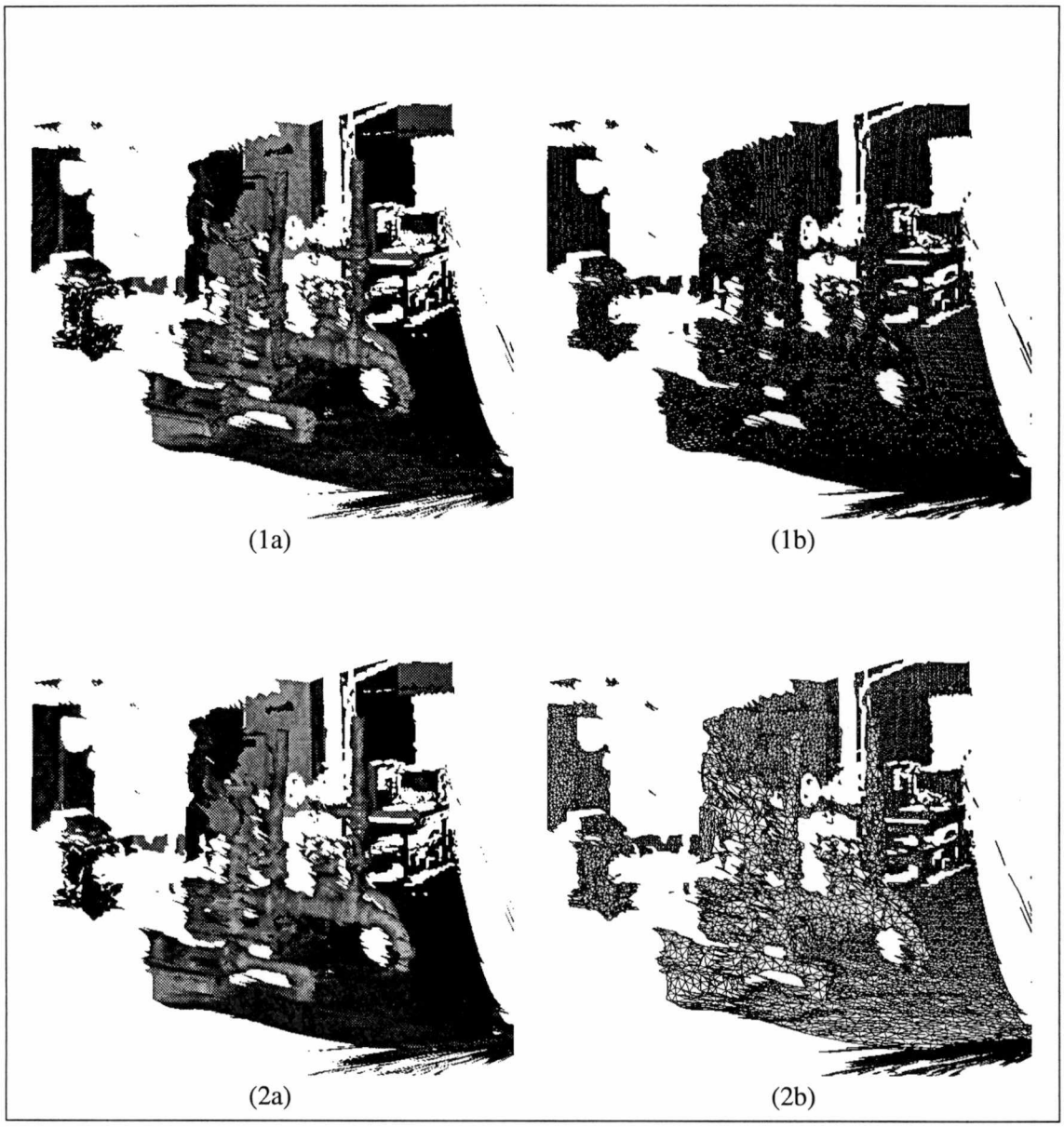


Figure 5.10: *(1a) Initial Perceptron model with 116,544 faces, (1b) wire-frame of the initial Perceptron model, (2a) Perceptron model after 72.6% reduction, and (2b) wire-frame of the 72.6% reduced model.*

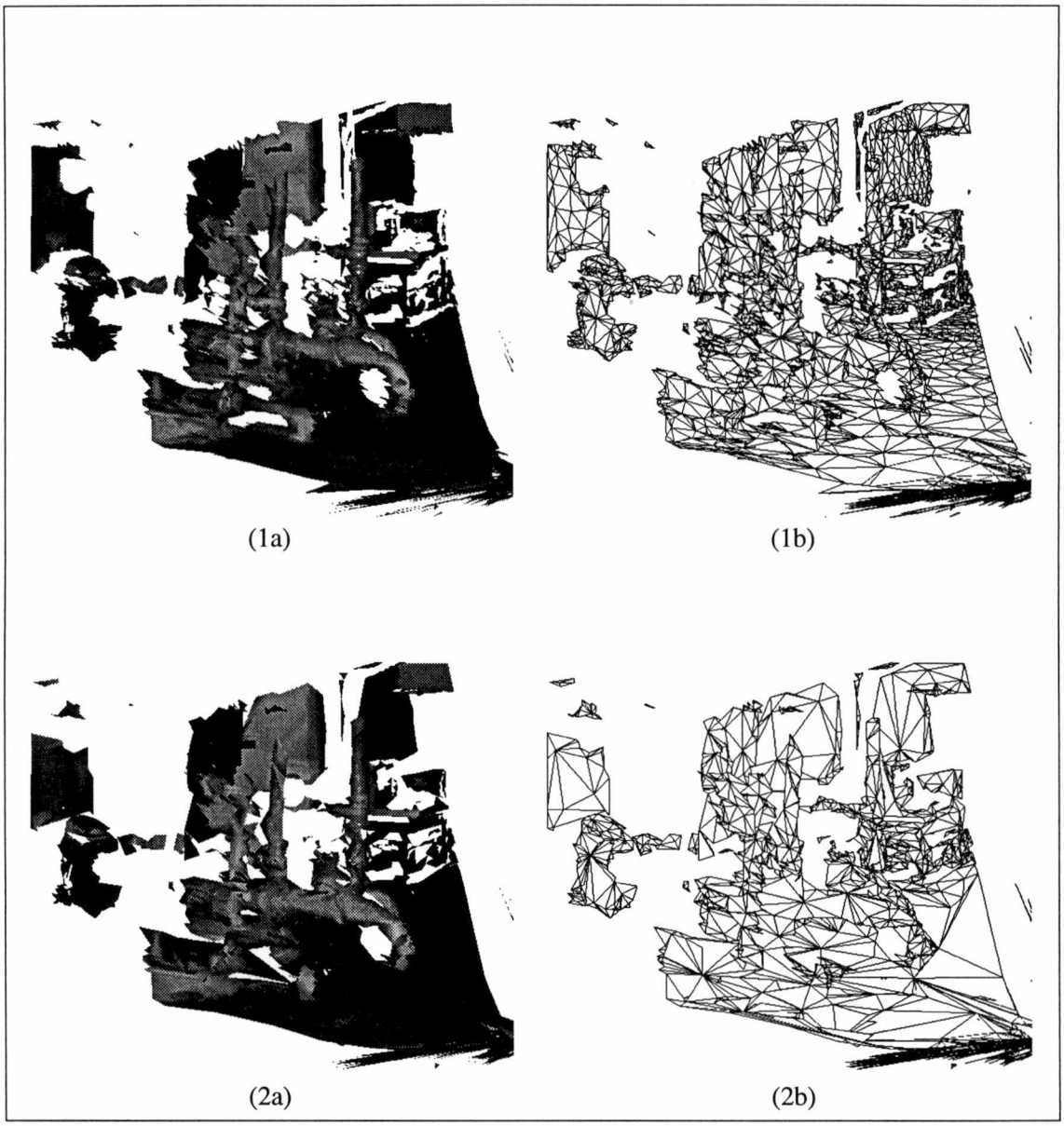


Figure 5.11: *(1a) Perceptron model after 91.5% reduction, (1b) wire-frame of the 91.5% reduced model, (2a) Perceptron model after 96.1% reduction, and (2b) wire-frame of the 96.1% reduced model.*

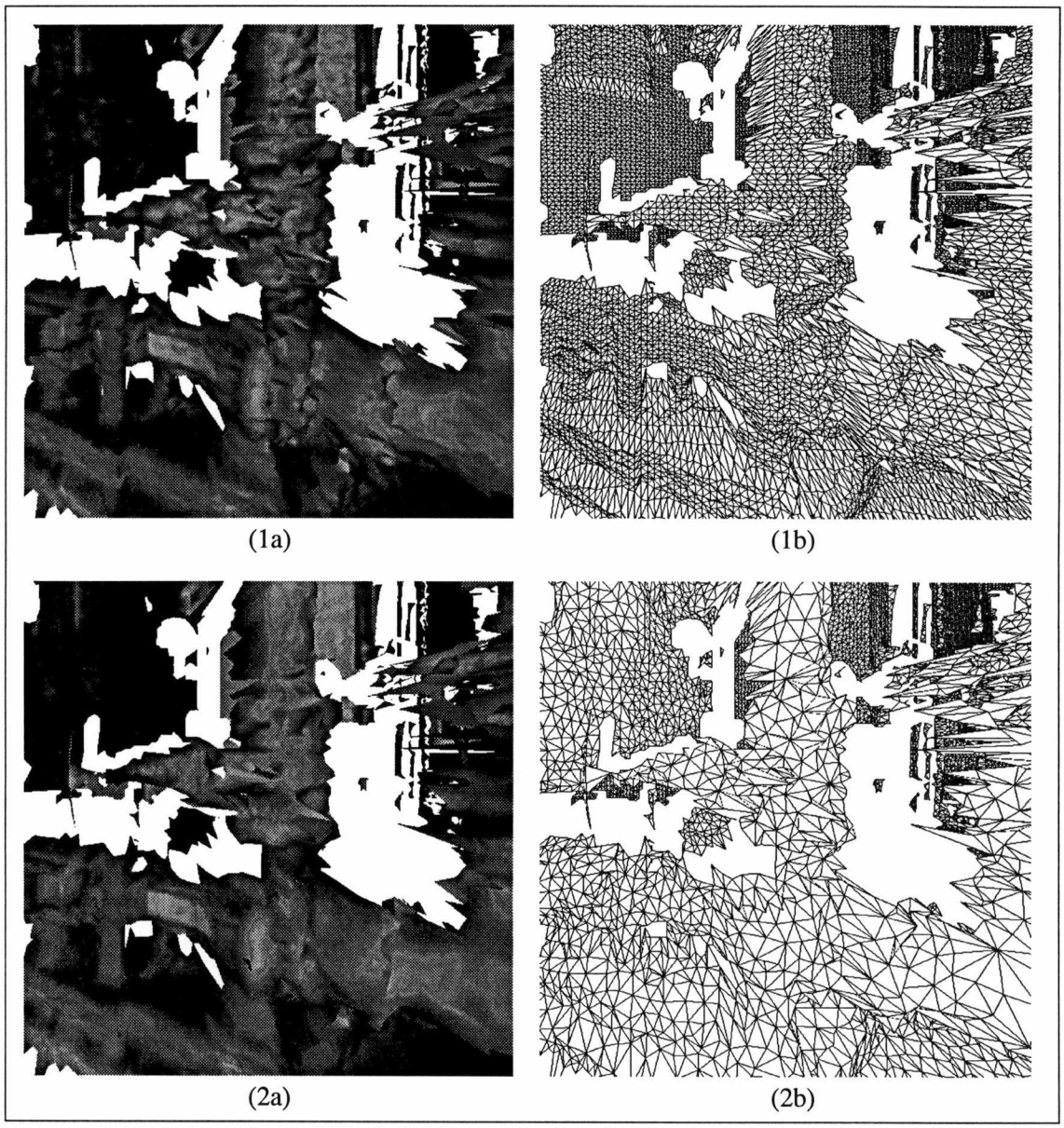


Figure 5.12: *(1a) Zoomed view of the initial Perceptron model showing the pipe in the middle of the scene, (1b) wire-frame of the initial model, (2a) zoomed view after 62.5% reduction, and (2b) wire-frame of the 62.5% reduced model.*

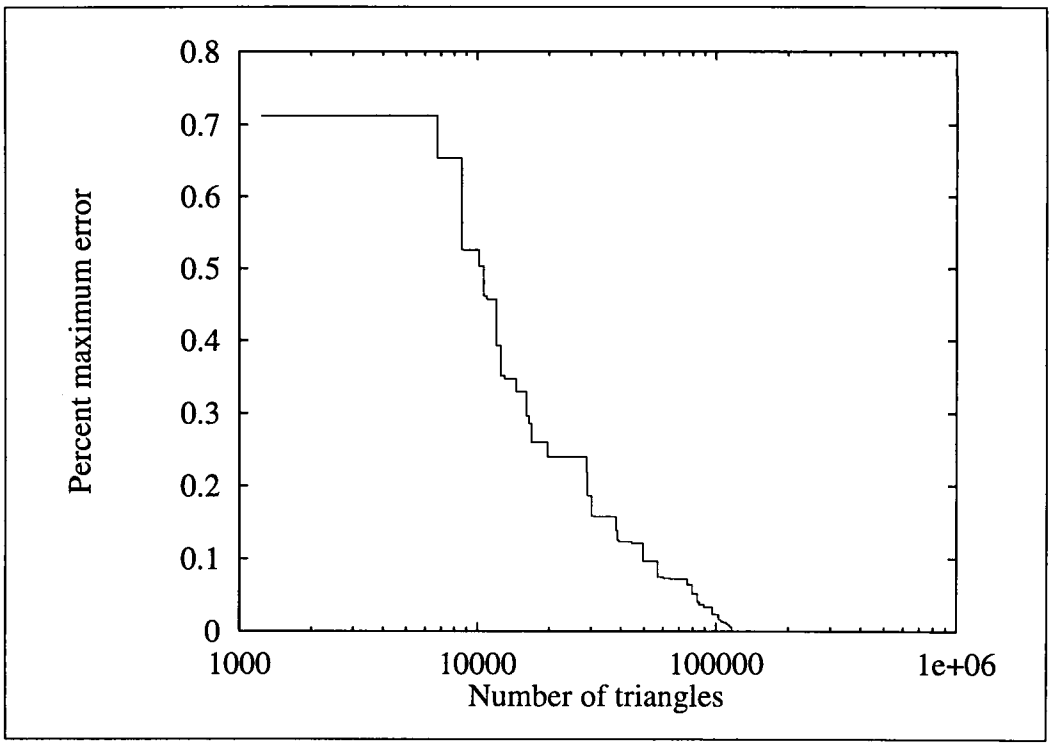


Figure 5.13: *Maximum calculated error versus the number of triangles for the real range image with segmentation information.*

aid in reduction. The resulting weights used for this model are 50.0 for geometry, 1.0 for normal and curvature, 10.0 for color or segmentation, and 15.0 for boundary. Figure *5.13* shows a graph of the error versus the number of triangles in the model. For this model also, the error increases exponentially as the number of triangles is reduced.

As can be seen in the figures, the data set reduces rather well. The pipes are still visible and flat surfaces, such as the floor and walls, are composed of few triangles. The reduction does smooth some of the noise present in the initial data, but spikes at the bottom of the scene are still prevalent. Reduction much past 98% results in the loss of the structure contained in the scene, which is to be expected given the large number of objects contained in this complex scene.

5.2.3 Perceptron Range Data with Color Texture Mapped Image

The data set presented here is created with a range image taken with the Perceptron as well (see Figure *5.14*). Here, however, we are given a registered color intensity image, shown in Figure *5.14*

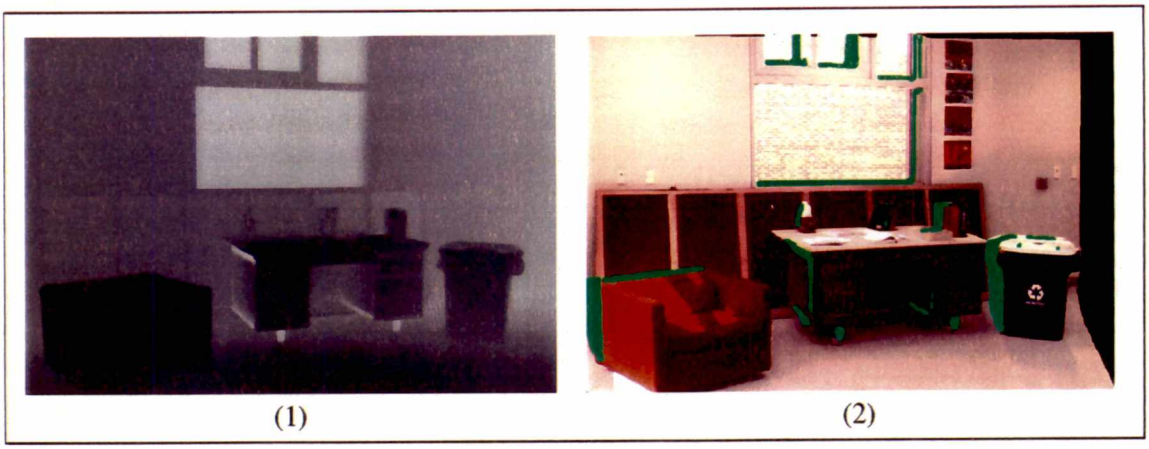


Figure 5.14: *(1) 768x511 range image taken with the Perceptron laser range camera, and (2) a registered color image.*

to accompany the range image. Details of the registration process are covered by Elstrom [28]. Here the color image is used for the color vector as a parameter in the reduction. In this manner, the reduction should be limited across color boundaries and regions of like color should be listed higher in the reduction hierarchy. The green areas in the color image are areas which were visible from the laser scanner, but not the color camera.

One very noticeable result of using the color information as one of the parameters in the reduction is visible on the wall viewed through the window located outside the room. The texture pattern of the brick on the wall results in almost no reduction in that area. As a result, other areas containing colors which are more alike get reduced much more, such as the wall located inside the room. Once again, using the Perceptron data, holes are present in the data. This is especially noticeable in shadowed areas along the bottom half of the wall behind the desk and the top of the desk which does not return good range values due to the angle of incidence with the laser. With the large amount of reduction, this results in the holes growing. The initial weights chosen for the reduction are 50.0 for the geometry and color, 25.0 for the boundaries, and 1.0 for the normal and curvature.

As can be seen in Figures *5.15* and *5.16* the uniformly colored surfaces are greatly reduced. The lack of reduction in the brick wall did, however, cause areas such as the desk to be reduced further than desired. Also, holes in the data in areas not visible to the laser caused the area behind

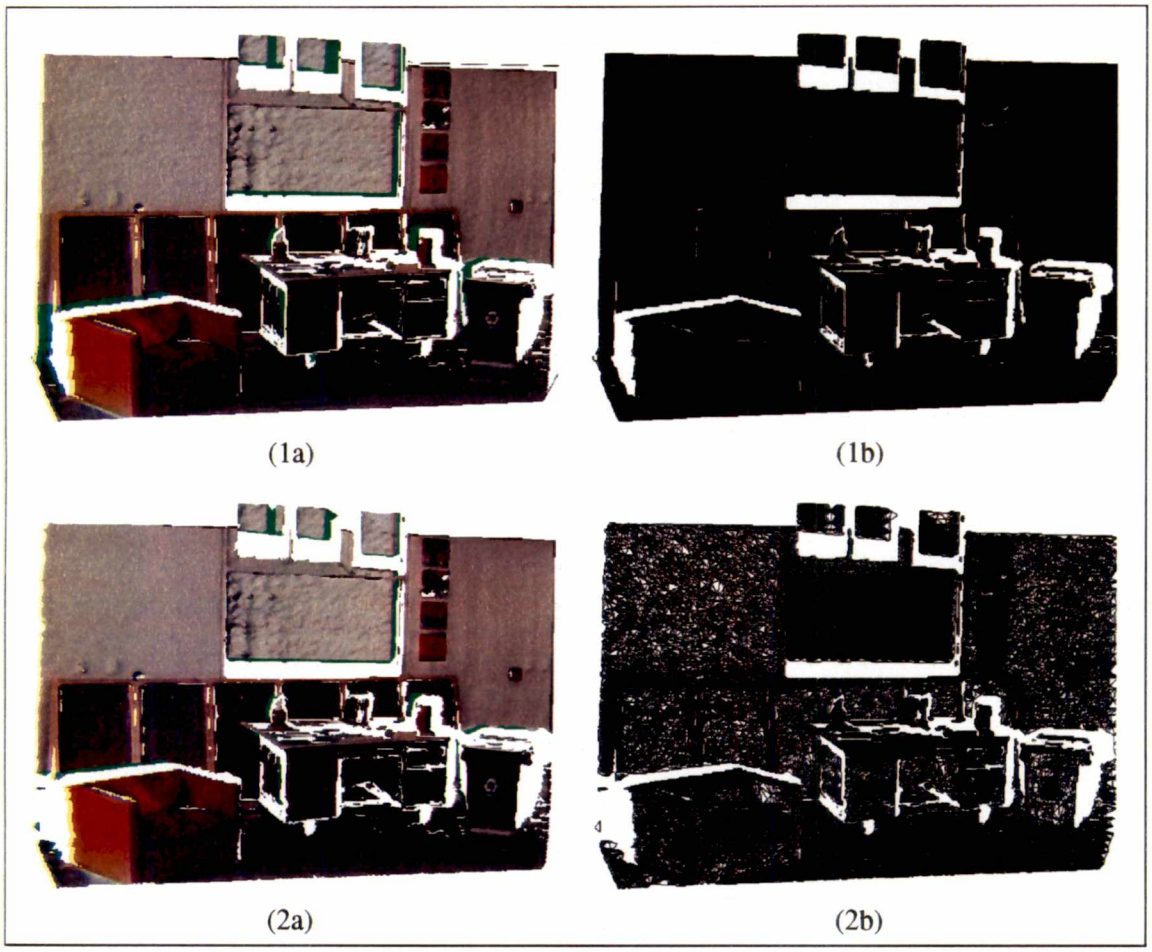


Figure 5.15: *(1a) Initial color model with 179,984 faces, (1b) wire-frame of the initial color model, (2a) color model after 60.4% reduction, and (2b) wire-frame of the 60.4% reduced model.*

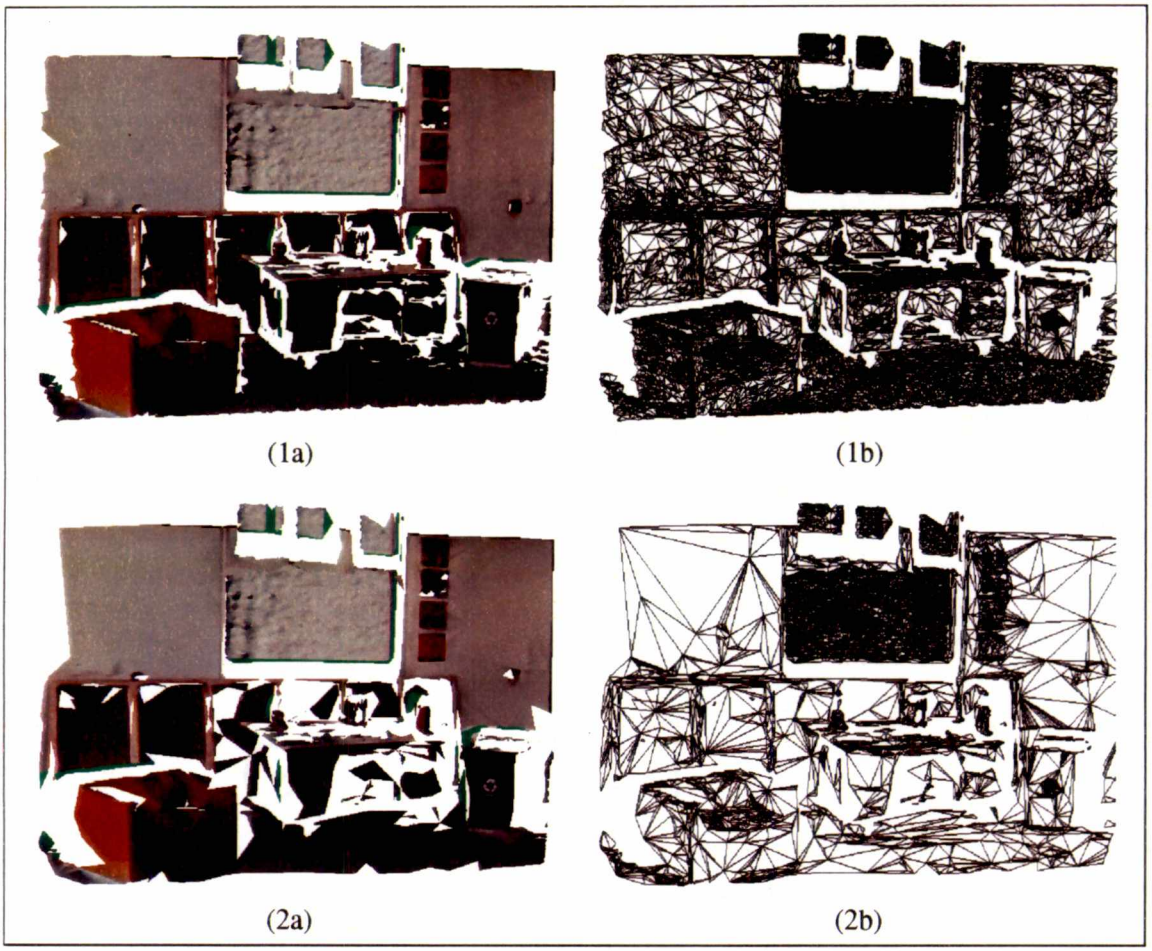


Figure 5.16: *(1a) Color model after 80.3% reduction, (1b) wire-frame of the 80.3% reduced model, (2a) color model after 94.2% reduction, and (2b) wire-frame of the 94.2% reduced model.*

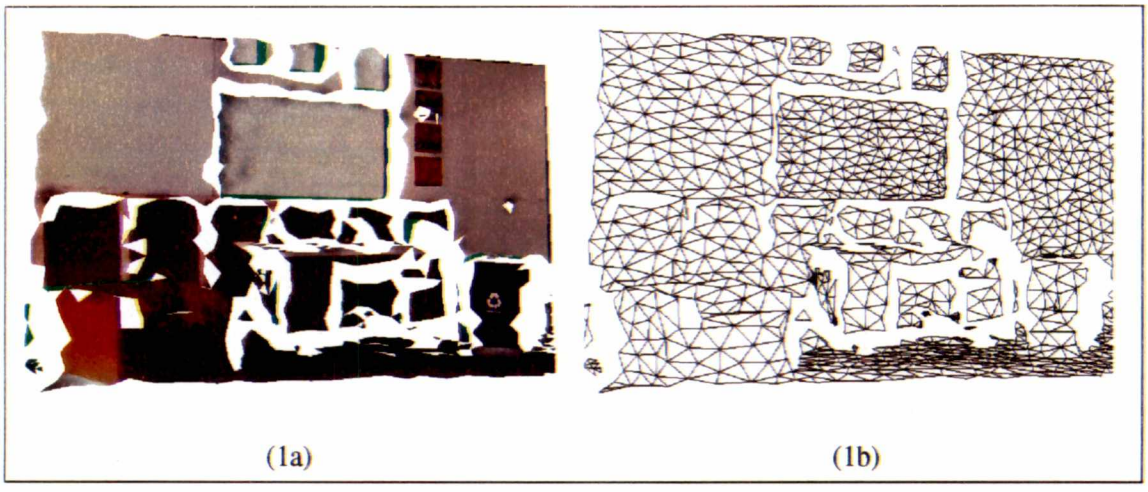


Figure 5.17: *(1a) Color model after 99.0% reduction without using color information, and (1b) wire-frame of the 99.0% reduced model.*

the desk to be reduced more than desired. Other interesting features include the pictures on the wall and the symbol on the recycle bin. These areas of contrasting color also were not reduced due to the high color weight. To see the effect of the color vector, the model was reduced again with the color weight set to 1.0. The result is shown in Figure *5.17*. Here the model is reduced by 99%. Without the color information effecting the reduction as much, the reduction is much more even across the entire scene. The color, in this case, would not have been such a factor if it were not for the brick wall located behind the scene. This model gives a good example of how the color in the scene can effect the reduction. Figure *5.18* shows the error incurred during the reduction of the model. A large jump in the error occurs around 100,000 triangles. Once again, the error function we are using gives the maximum calculated error. This jump is the results of a reduction which introduced a large error in one triangle in the model. The subsequent reductions produce much smaller errors. Thus the maximum error remains constant for several iterations during the reduction. The error metric we have chosen has no effect on the selection of the edge to be collapsed. Therefore situations such as this one can occur. In this case, the edge collapse that introduces the error occurs on the arm of the chair.

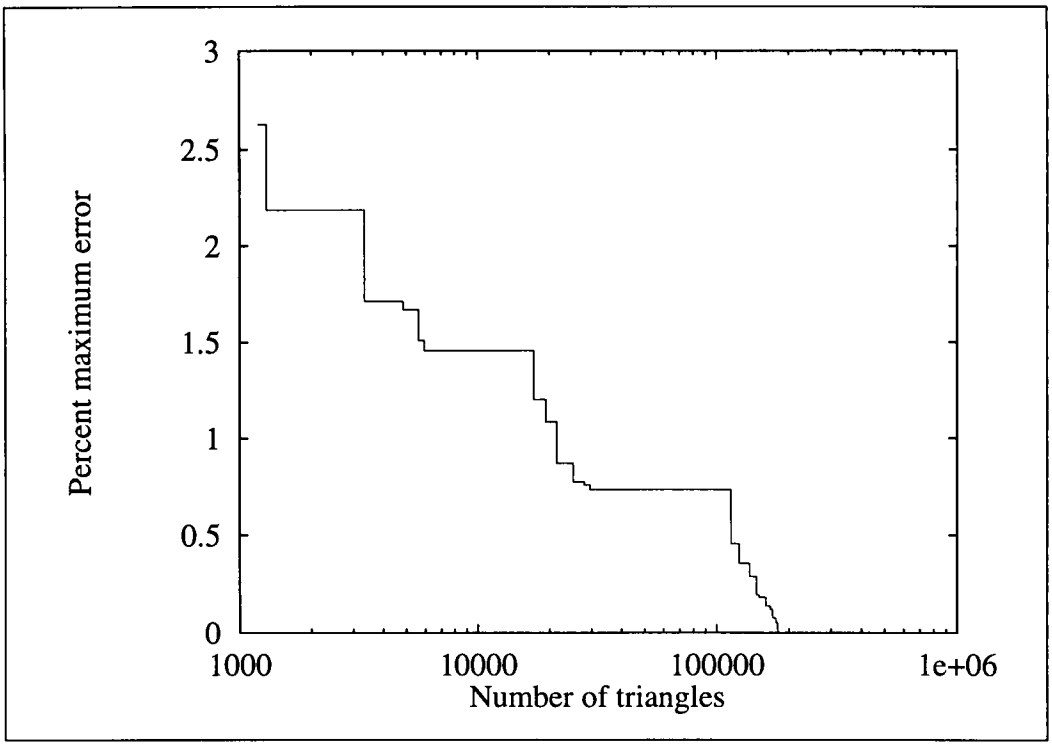


Figure 5.18: *Maximum calculated error versus the number of triangles for the full 3D model created from 12 range images.*

5.2.4 Coleman Range Data with Confidence

The data set presented here, supplied by Oak Ridge National Laboratory, is created from a range image taken by a Coleman laser range camera. This range image has much less noise than the previous two range images presented, but does not have an accompanying intensity image (see Figure *5.19*). As a result of less noise, fewer number of holes are present when the model is created. The data does however have a confidence value associated with each point in the scan. This value is mapped to a color and subsequently used as a factor in the mesh reduction. The data with the highest confidence is colored red and the lowest confidence is blue. The vector weights used for this reduction are 20.0 for geometry, 0.1 for normal, 2.0 for color, 10.0 for boundary, and 0.25 for curvature. The areas with the lowest confidence are the ones with the most noise, as thus, the most holes in the model. The results of the reduction are better than the previous models created using range images from the Perceptron scanner simply due to the fact that there are not as many holes in the initial model (see Figure *5.20* and Figure *5.21*). In this case, the color does not

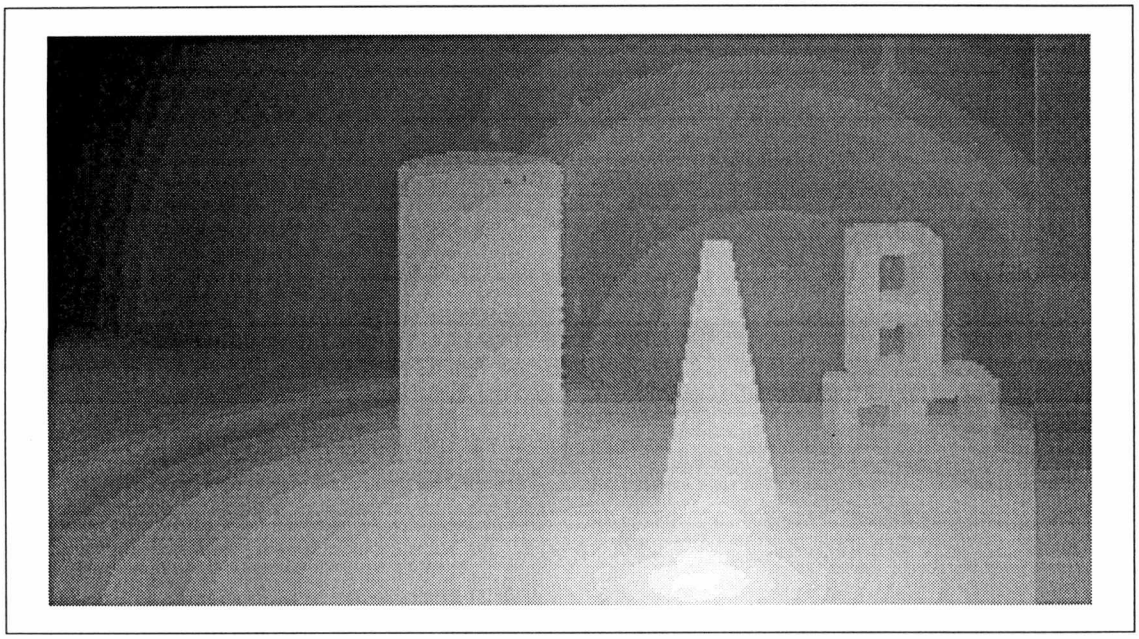


Figure 5.19: *335x181 range image taken with the Coleman laser range camera.*

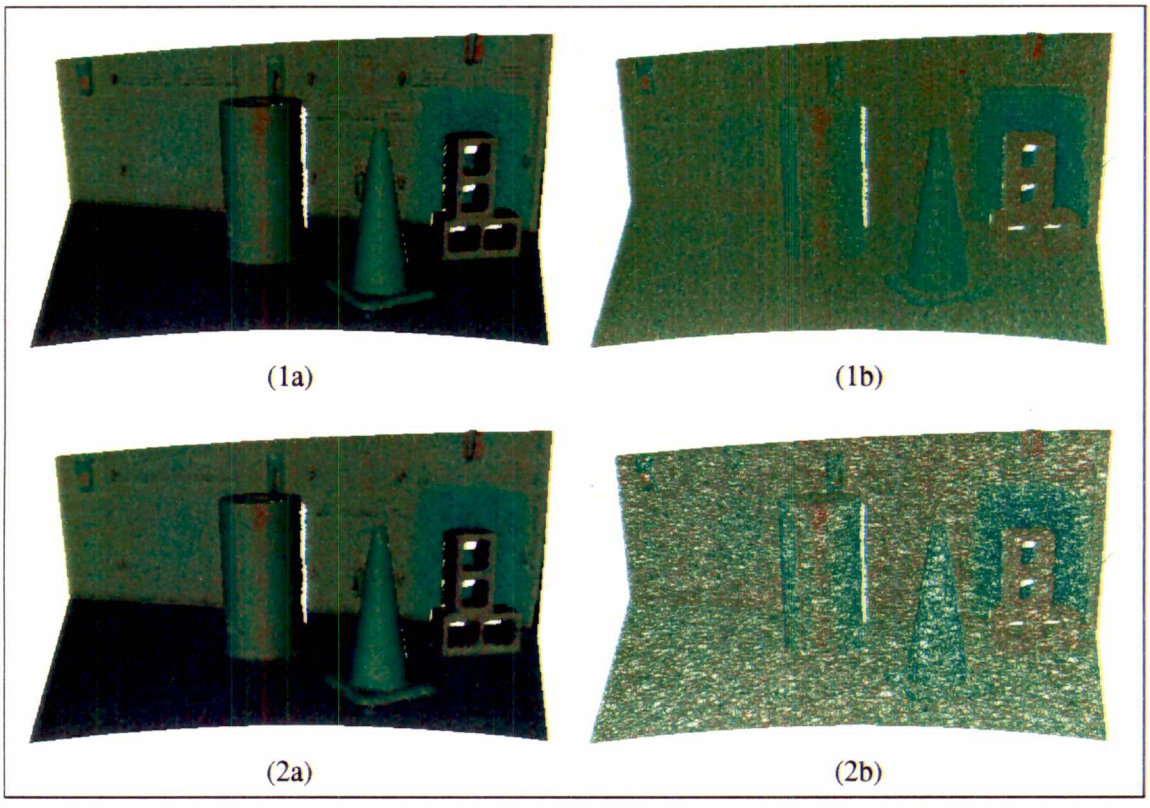


Figure 5.20: *(1a) Initial coleman model with 118,996 faces, (1b) wire-frame of the initial coleman model, (2a) coleman model after 70.4% reduction, and (2b) wire-frame of the 70.4% reduced model.*

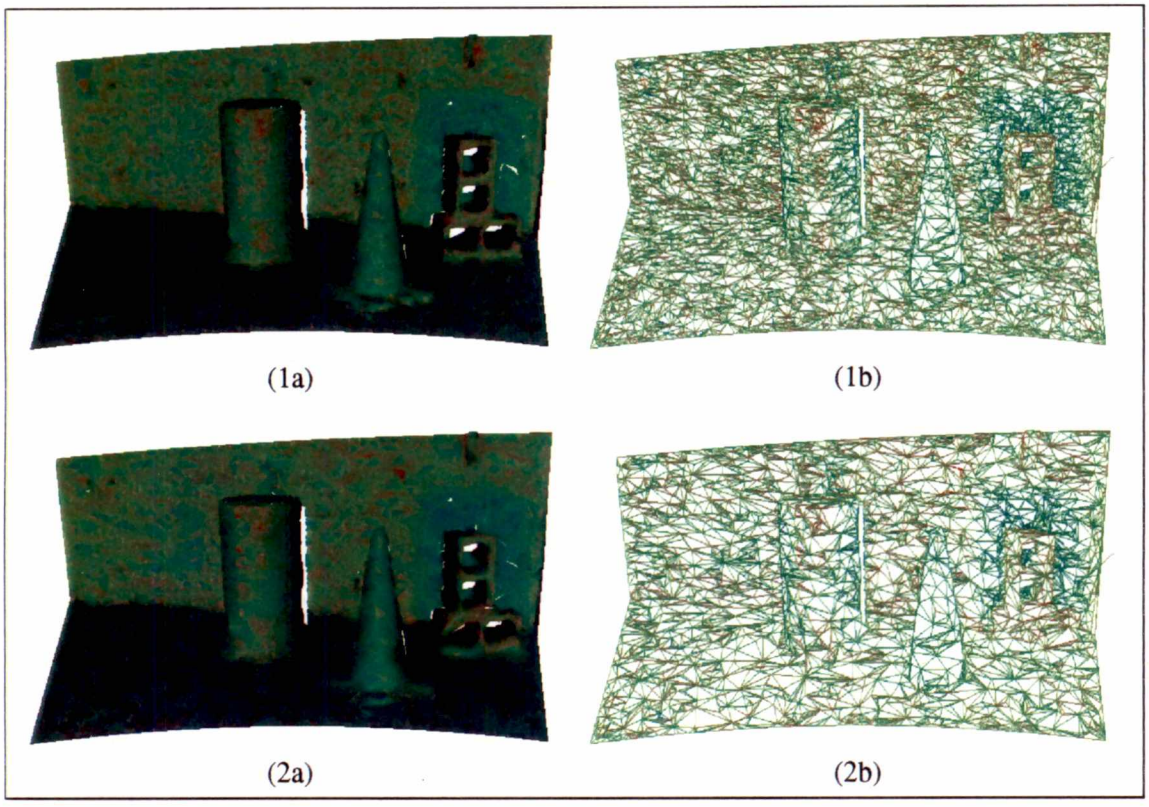


Figure 5.21: *(1a) Coleman model after 90.2% reduction, (1b) wire-frame of the 90.2% reduced model, (2a) coleman model after 94.7% reduction, and (2b) wire-frame of the 94.7% reduced model.*

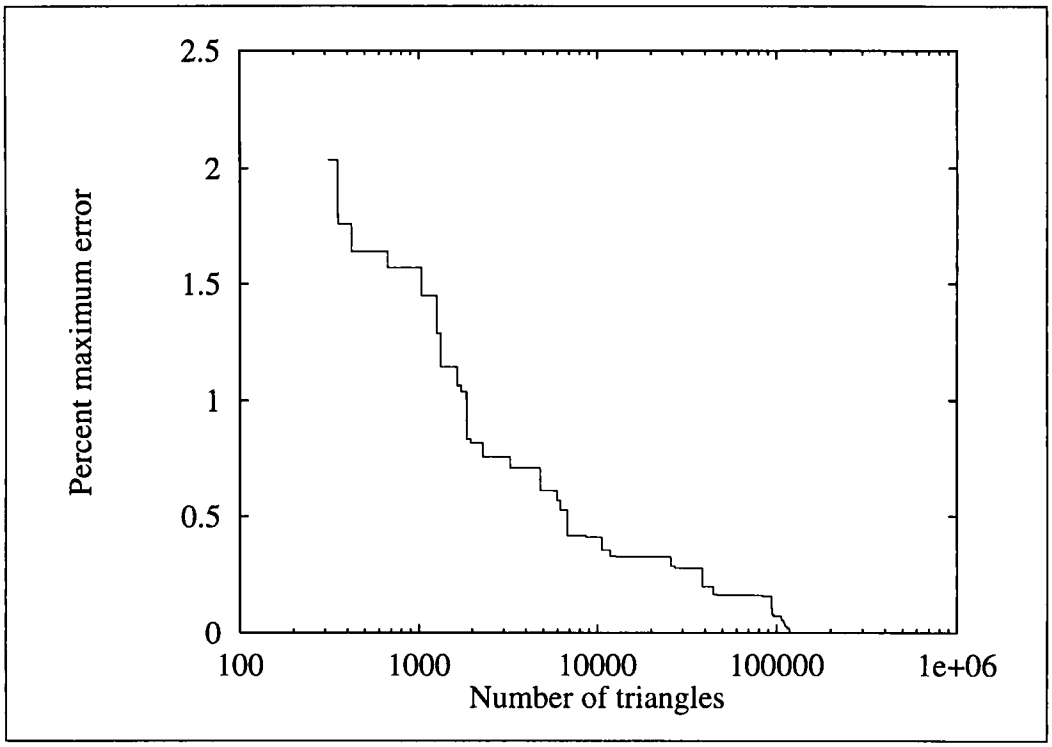


Figure 5.22: *Maximum calculated error versus the number of triangles for the full 3D model created from 12 range images.*

aid much in the reduction of the model due to the randomness across the image of the confidence measure, therefore the weight for color is not as high as the other models using color as a feature. But even at 95% reduction, little difference is seen between the initial and reduced model. The error graph for this reduction is shown in Figure *5.22*.

5.2.5 Digital Elevation Map

This digital elevation map (DEM), shown in Figure *5.23*, is part of the GTOPO30, global 30 arc second elevation data set, developed by the U.S. Geological Survey's Earth Resources Observation Systems Data Center. The spacing of the data is approximately one kilometer per pixel. The data set we use is a portion of Southern Florida. The elevation is multiplied by 50 to increase the variation in the data. The reduction is performed with weights of 15.0 for geometry, 0.1 for normal, 2.0 for curvature, and 10.0 for boundary information. As can be seen in Figure *5.24*, the areas with the least variation in elevation are reduced first leaving the details where the terrain

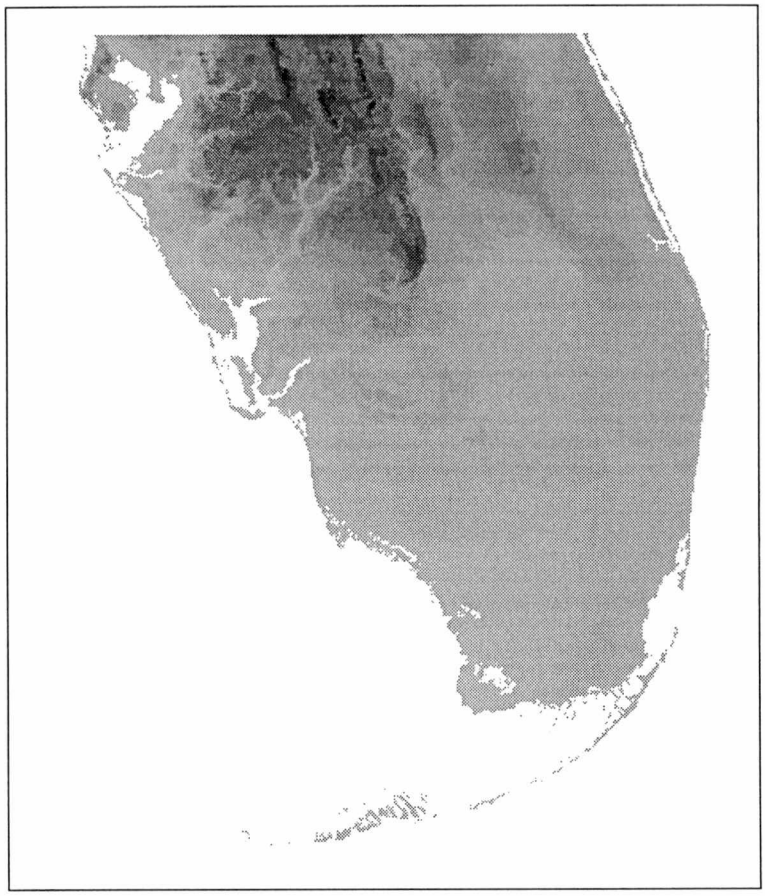


Figure 5.23: *360x440 digital elevation map of Southern Florida.*

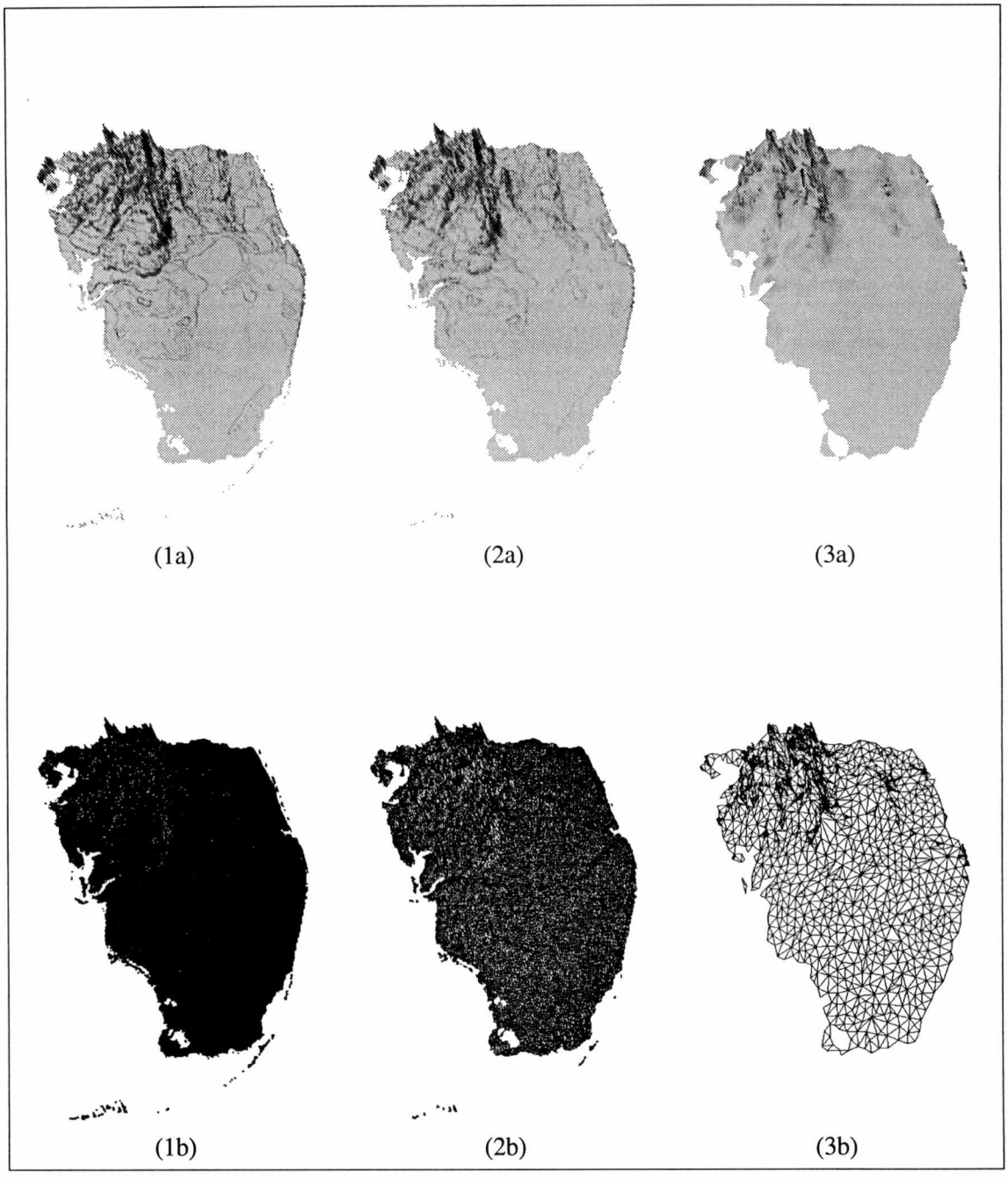


Figure 5.24: *(1a) Initial model created from DEM with 100,000 faces, (1b) wire-frame of the initial model, (2a) model after 77.0% reduction, (2b) wire-frame of the 77.0% reduced model, (3a) model after 96.0% reduction, and (3b) wire-frame of the 96.0% reduced model.*

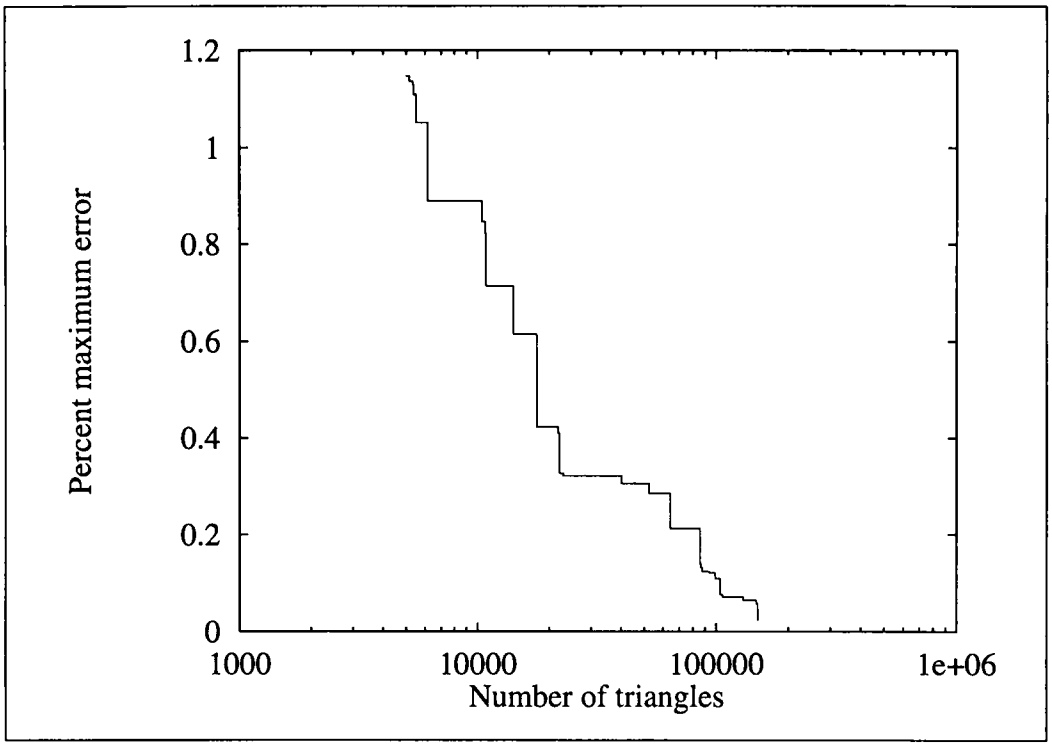


Figure 5.25: *Maximum calculated error versus the number of triangles for the DEM model.*

varies. It should be noted that some of the smaller islands in the initial scene are not present in the reduced model. As edges are collapsed and faces deleted, new boundary edges are created and eventually these can actually pinch off from the model or, if a part of the model is small enough, actually be removed from the model. This is not viewed as a problem in the algorithm since this occurs at lower resolutions and the details can be viewed when needed. The error graph for this reduction is shown in Figure *5.25*.

5.2.6 Fused Range Data Sets

The model presented here is the result of fusing 12 different range images using occupancy grids as presented by Elsner [27]. The full 3D model is the result of running the marching cubes algorithm on a 3D occupancy grid. By using multiple range images, the large spikes evident in the previous models are filtered and the holes present from having only one viewpoint are filled. The pattern vector reduction was run using a weight of 4.0 for the geometry, 1.0 for the normals, and 3.0 for the curvature. In this case, color has no effect since each vertex has the same color. Also, boundary

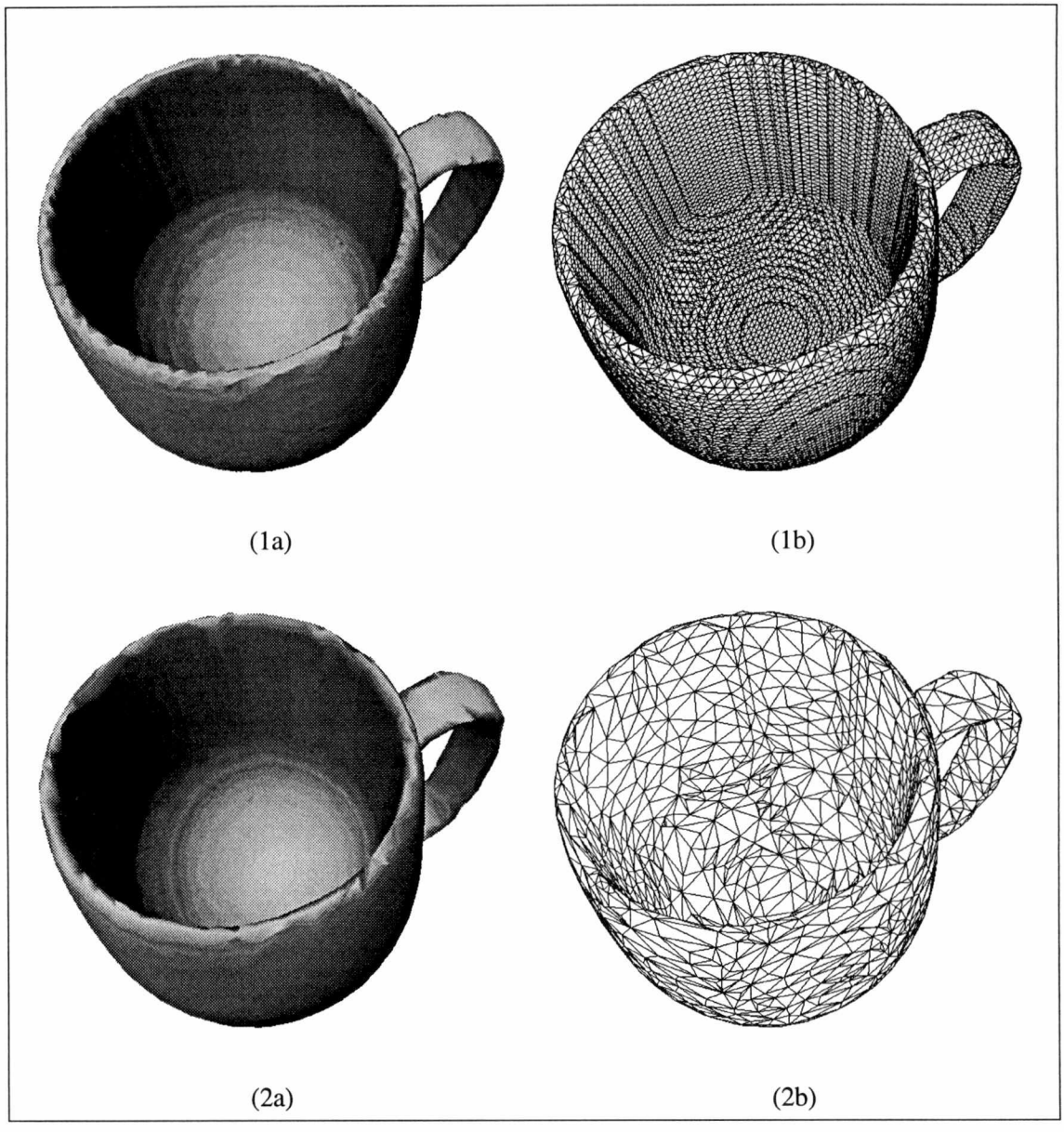


Figure 5.26: *(1a) Initial mug model with 38,268 faces, (1b) wire-frame of the initial mug model, (2a) mug model after 89.9% reduction, and (2b) wire-frame of the 89.9% reduced model.*

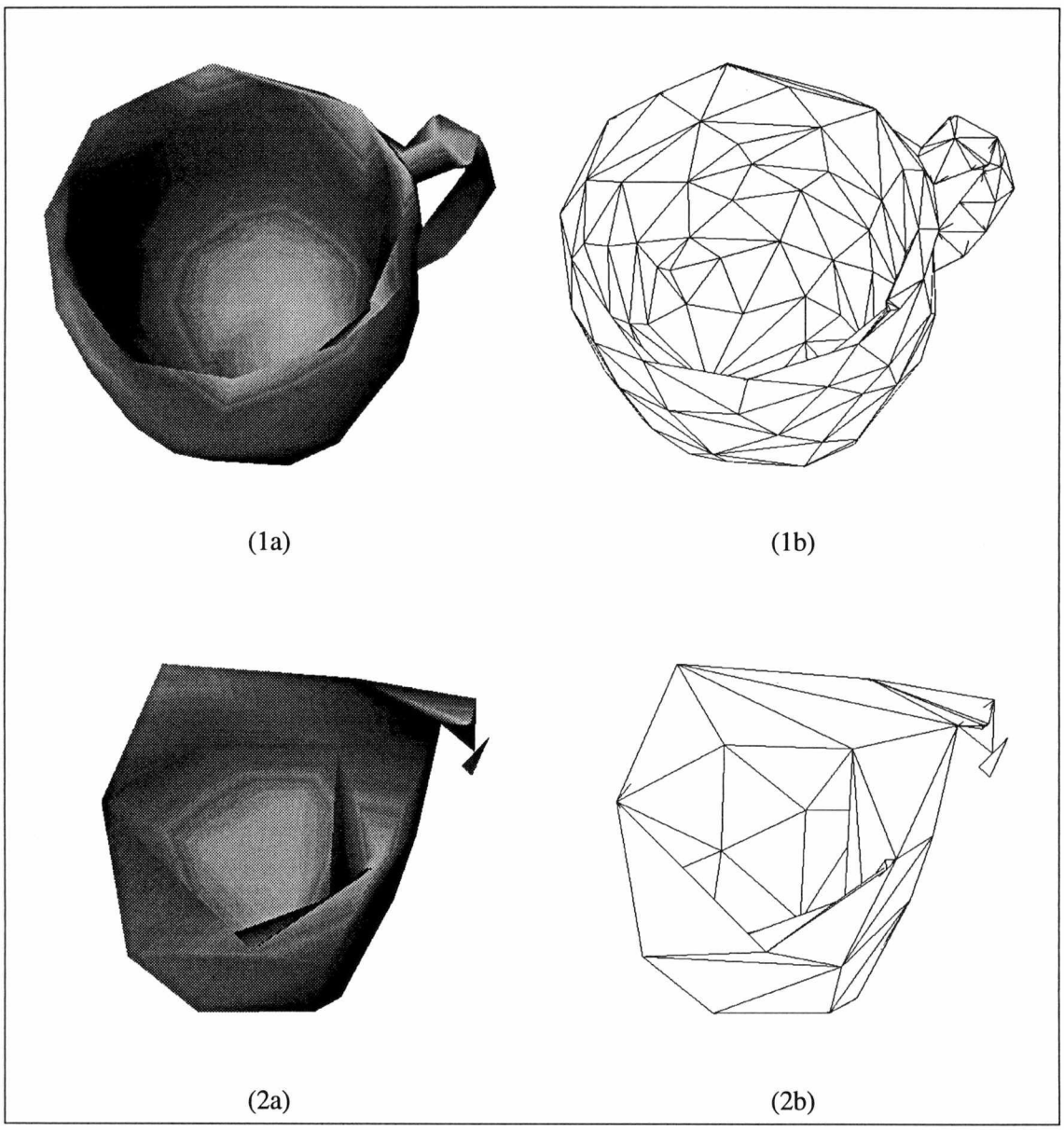


Figure 5.27: *(1a) Mug model after 99.1% reduction, (1b) wire-frame of the 99.1% reduced model, (2a) mug model after 99.8% reduction, and (2b) wire-frame of the 99.8% reduced model.*

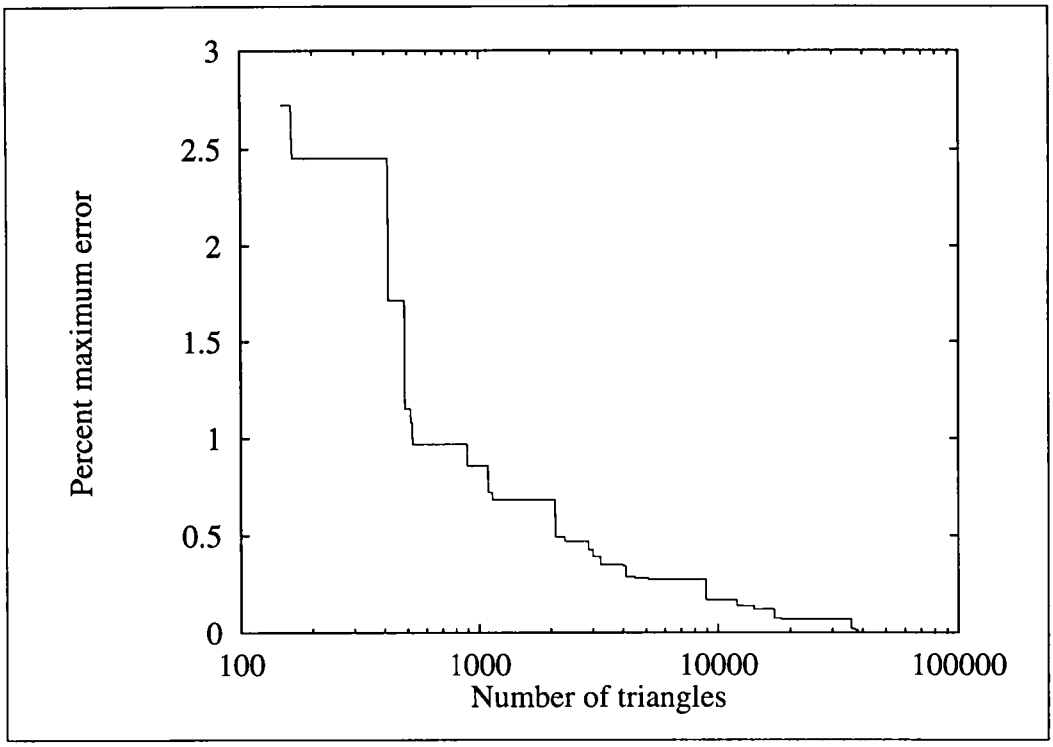


Figure 5.28: *Maximum calculated error versus the number of triangles for the full 3D model created from 12 range images.*

information has no effect since there are no boundaries in the final closed object.

The results of the reduction are shown in Figure *5.26* and Figure *5.27*. The reduction took about 30 seconds running on a Silicon Graphics Indigo2 with a 195MHz R10,000 processor and 128MB of RAM. As can be seen, the mug is still very much perceptible even with high reduction. With only 10% of the initial triangles, there is almost no difference in the appearance of the model. One interesting occurrence in this reduction is that the handle of the mug is pinched in two. This occurs in a similar manner to the islands being removed in the DEM example given in the previous section. The error graph for this example is shown in Figure *5.28*. Again, the exponential nature is evident.

5.3 Error Analysis

Certain trends are noticeable across all the data sets when viewing the results of the error graphs. Figure *5.29* shows the error results of all the reductions given in this chapter. The data

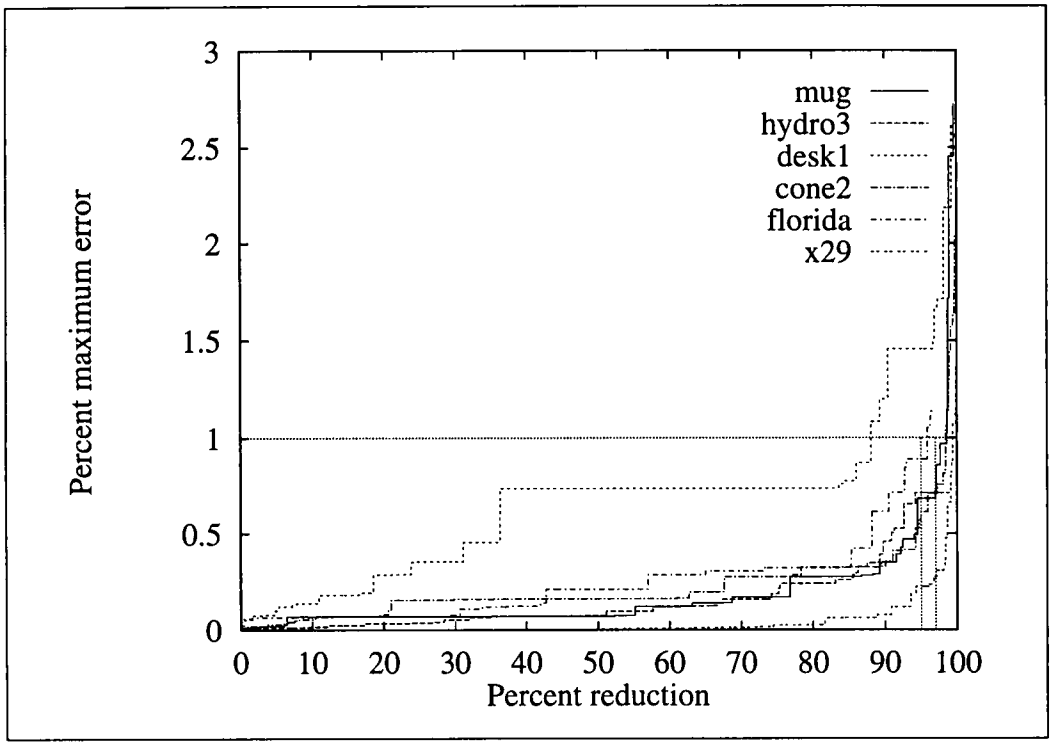


Figure 5.29: *Maximum calculated error for all models versus the number percent reduction.*

has been normalized based on bounding box size and the number of initial triangles. The graph shows that the data reduction for each model has almost the exact same percent maximum error as a function of percent reduction. The synthetic image has the least error, which is to be expected since it has no noise in the range data. The data set with the most noise, the desk image with the color texture map, likewise has the highest error. All the remaining error plots fall within a small tolerance of each other between the two extreme cases. Subjectively, good results are available below 1.0% maximum error which, from the graph, is around 95% to 97% reduction in most cases. As the reduction approaches 100% the model begins to become unrecognizable, which is to be expected. Some models are pinched in two or smaller disconnected portions removed at high levels of reduction. Once again, this is not considered a problem since the details of the model can be viewed when needed.

There is no good qualitative method to judge the visual quality of the reduced models since this judgment is subjective. We do, however, give results of subtracting views of the original and reduced models as described in Section *4.3.1* for each of the models reduced. The thresholded image in each case shows where pixel values have changed by more than 15% from the original. In each case, a reduced model of at least 94% is chosen to show the difference between a very high and very low resolution model. The first is the model created from synthetic range data shown in Figure *5.30*. One major source of error in this model is around the right front wing. Other small errors occur around the boundary of the model. The error from the first model created from Perceptron range data is shown in Figure *5.31*. Here most of the error occurs near holes created from shadows in the data. The error from the second model created from Perceptron range data is shown in Figure *5.32*. Here, as in the previous model, most of the error occurs near holes. In addition, some error long the left of the interior wall is visible. As can be seen by the images, almost no error is present on the brick wall outside of the room. Once again, this is a result of having the color weight value set high. The model created from the Coleman range data on the other hand shows almost no visual error (see Figure *5.33*). Figure *5.34* shows the error results of the model created from the topological data. Most of the error from the reduction of this model is from the removal of the small details in the upper left corner. Also, the islands which

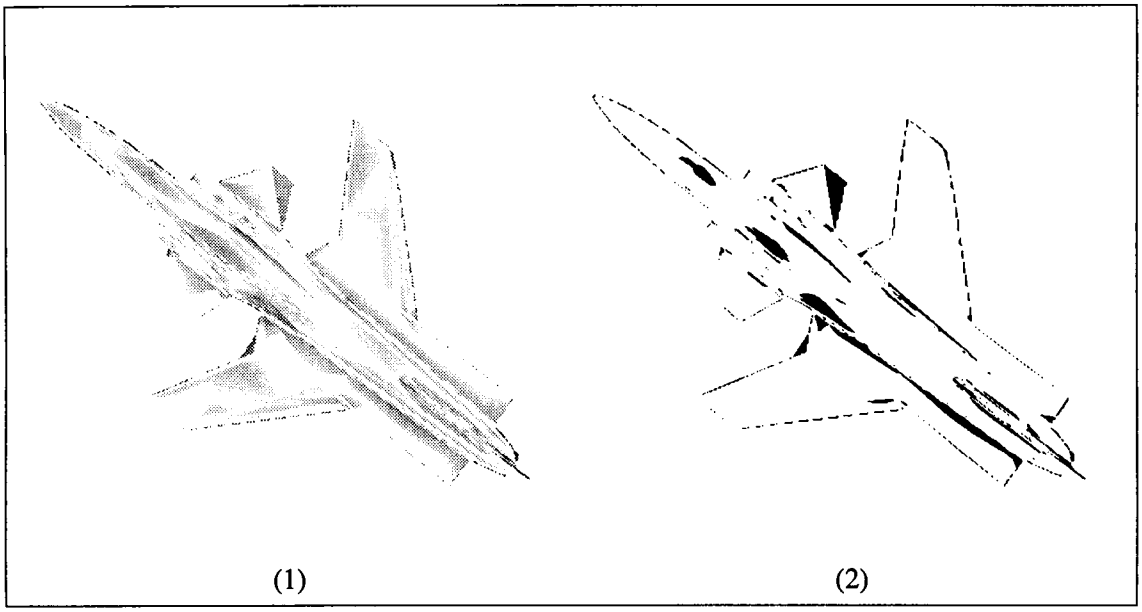


Figure 5.30: *(1) Visual error between the initial model created from synthetic range data and the 97.8% reduced model, and (2) visual error image thresholded.*

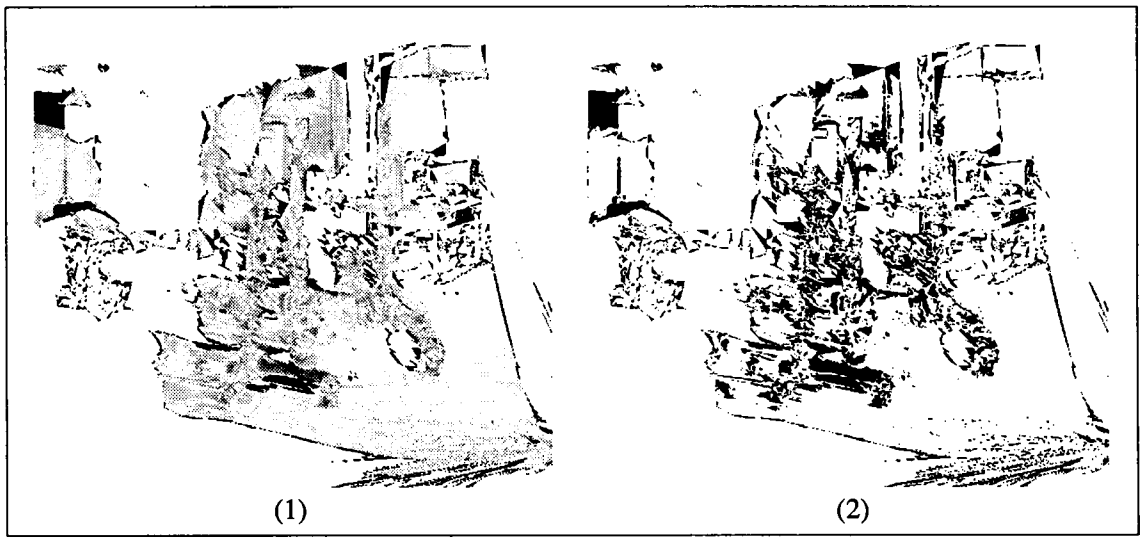


Figure 5.31: *(1) Visual error between the initial model created from Perceptron range data and the 96.1% reduced model, and (2) visual error image thresholded.*

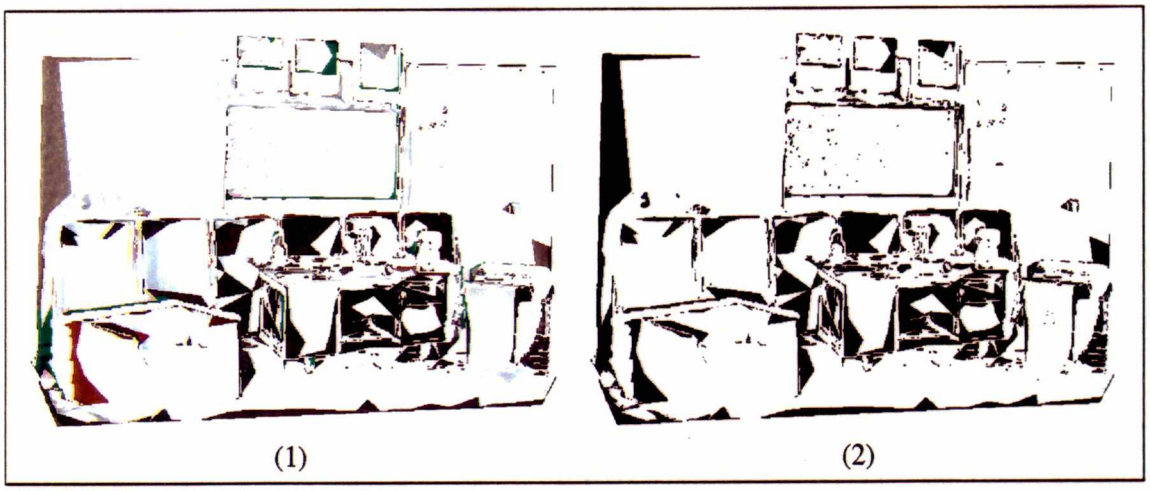


Figure 5.32: *(1) Visual error between the initial model created from Perceptron range data and color imagery and the 94.2% reduced model, and (2) visual error image thresholded.*



Figure 5.33: *(1) Visual error between the initial model created from Coleman range data and the 94.7% reduced model, and (2) visual error image thresholded.*

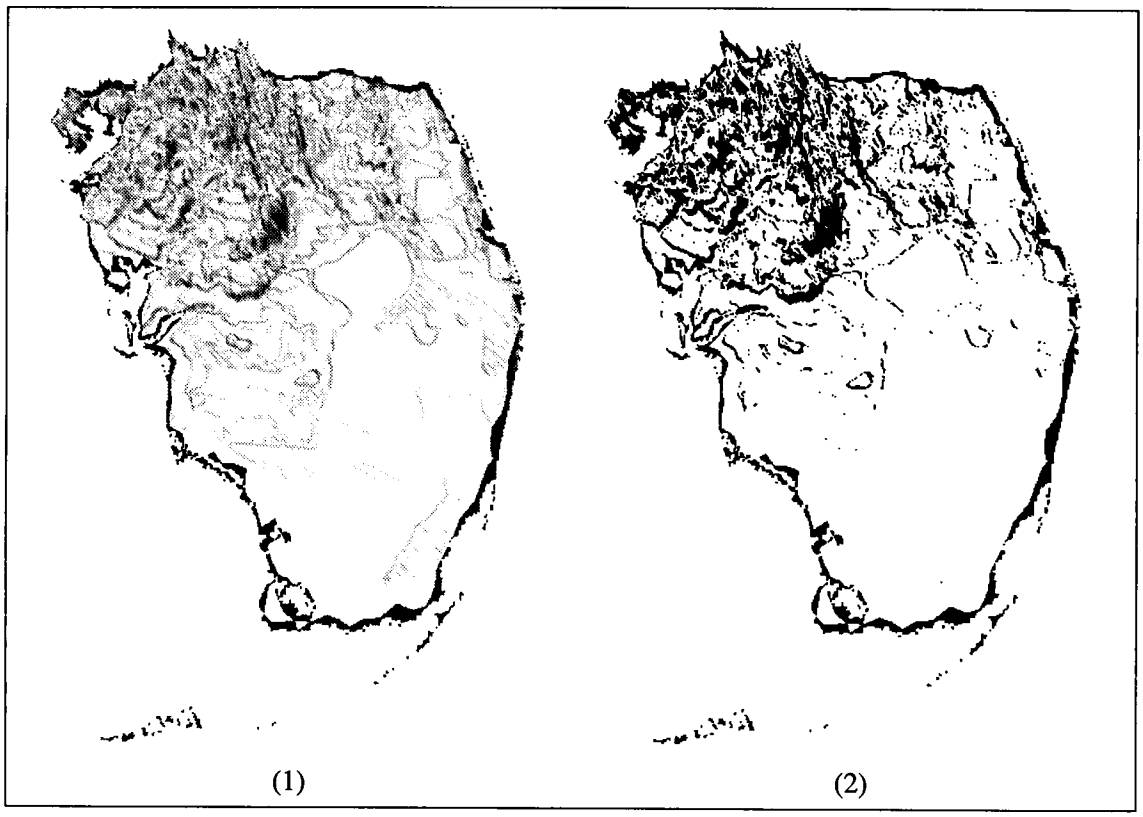


Figure 5.34: *(1) Visual error between the initial model created from DEM data and the 96.0% reduced model, and (2) visual error image thresholded.*

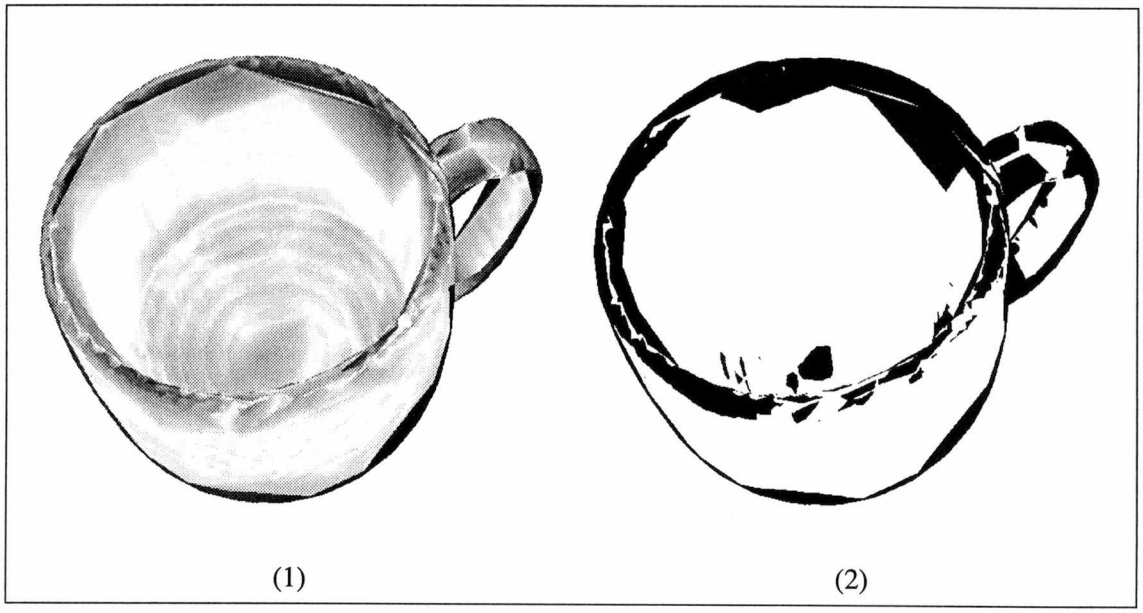


Figure 5.35: *(1) Visual error between the initial model created from multiple range data sets and the 99.1% reduced model, and (2) visual error image thresholded.*

were removed as a result of the reduction show up as errors. The final model tested is that of the reconstructed mug. The error shown in Figure *5.35* is from a 99.1% reduced model and as such is expected to show more error. Most of the error here is located around the top of the mug and along the handle.

5.4 Automatic Resolution Selection for Constant-Rate Interactivity

The automatic resolution selection is achieved by using the SoMRFaceSet Open Inventor node described in Section *4.2.2*. At this point, all the reduction has been performed off-line and the results stored as a multiresolution model. To maintain a constant frame rate, the subclass must know about the viewer used for display. This node has a timer with hooks, which when connected to a viewer, obtains information on update rates. This timer measures how many frames have been drawn in one second. With this information and knowing the number of triangles in the model, the number of triangles drawn per second is known. A control loop is created which automatically calculates the resolution of the model needed to maintain a constant frame rate from the number

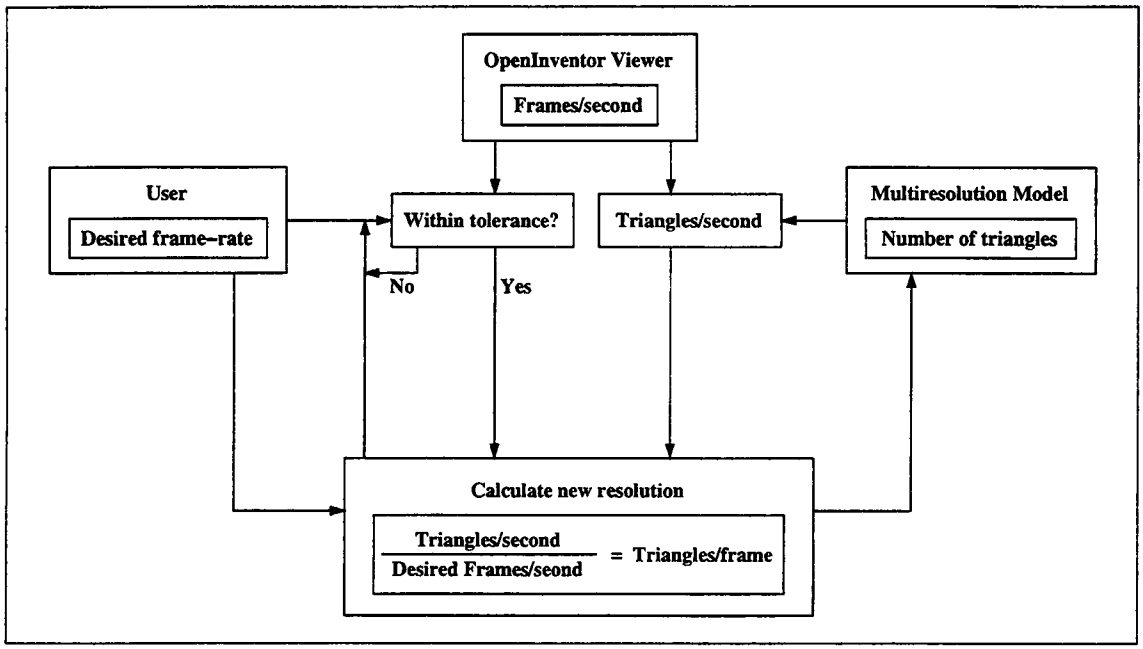


Figure 5.36: *Control loop to calculate the needed resolution to maintain a constant display rate.*

of triangles in the current resolution of the model, the number of frames being drawn each second, and the user's desired frame-rate (see Figure *5.36*). Once this is known, the timer changes the model to the appropriate resolution to obtain the frame rate desired by the user. In this manner, while the object is being moved, a lower resolution, higher frame rate model can be used, and once the object is stationary, the details of the model filled in. By keeping a copy of the initial model and the reduced model, the pointer to the data for the viewer can be quickly swapped when the user interacts with the model. Here our goal is speed of transition between the high and low resolution model and not the amount of memory used by the model, therefore, two copies of the model are kept. As previously stated, the reduction and creation of multiresolution models were tested on two machines. With the models created off-line, the constant interactivity can be run on any machine with Open Inventor. The models were displayed on a variety of machines, including a Silicon Graphics O2 and older Silicon Graphics Indigo2's. All machines quickly calculate the appropriate resolution needed to maintain a fixed frame-rate and use that resolution when moving the model. Figure *5.37* shows the results of displaying a multiresolution model using a standard

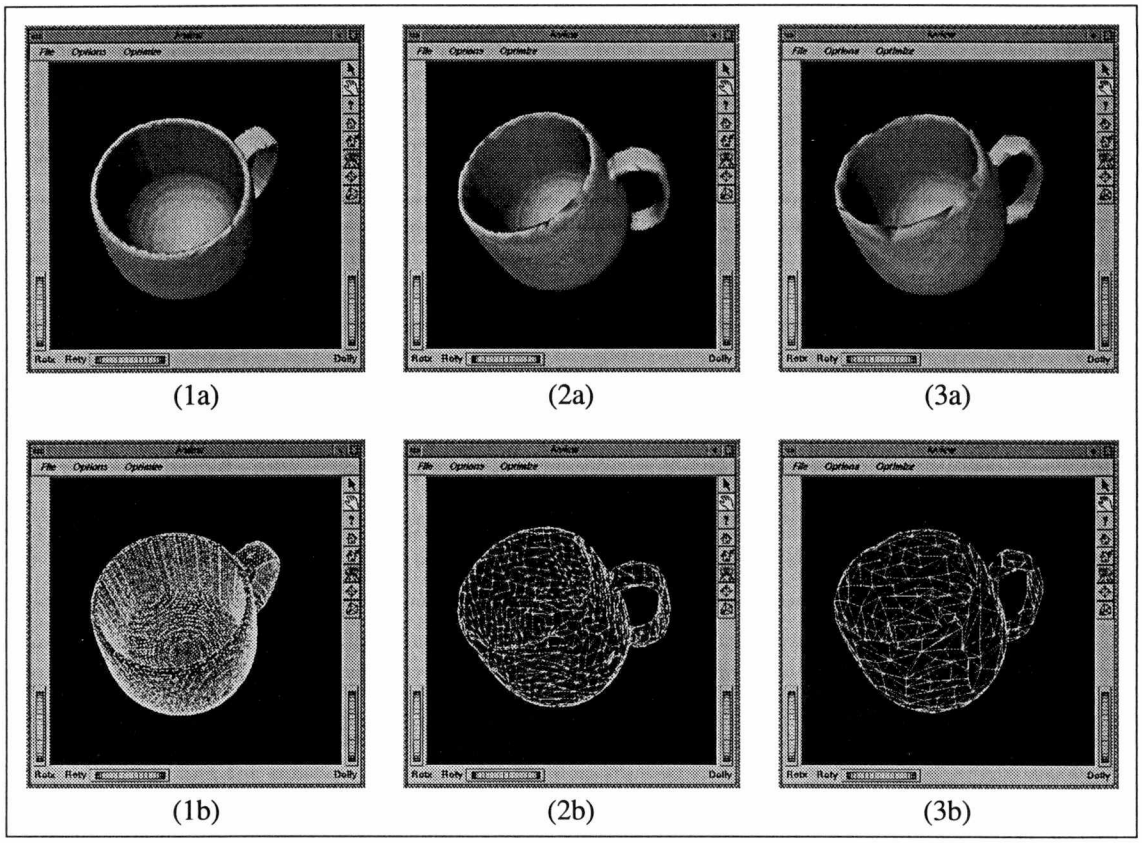


Figure 5.37: *Automatic resolution selection on performed using two separate machines: (1) the original model containing 38,268 triangles, (2) the resolution selected on a faster machine to maintain 60 frames per second (3,717 triangles), and (3) the resolution selected on a slower machine to maintain the same frame-rate (1,200 triangles).*

Open Inventor viewer. This example was run on a SGI Indigo2 MaxImpact and also on an older SGI Indigo2 XZ. The first movement of a multiresolution model in a viewer results in a slight delay of one to three seconds while the resolution is being adjusted. This is due to the dendrogram implementation which uses a linked list as described in Section *4.2.2*. After this initial delay, any movement of the model results in the resolution being reduced instantaneously to that required to maintain the given frame-rate. The original model contains 38,268 triangles. The faster machine automatically reduced the model to 3,717 triangles (90.3% reduction) to maintain 60 frames-per-second. The slower machine reduced the model to 1,200 triangles (96.9% reduction), 67.7% fewer then the faster machine, to maintain the same frame-rate.

5.5 CAVE System with Automatic Reduction

The CAVE System works very similar to any other viewer. The difference with the CAVE is that there are four Open Inventor viewers instead of one. To handle this, only one of the viewers is hooked into the SoMRFaceSet node, specifically, the viewer used to interact with the model. The number of triangles for the desired frame-rate is calculated and the model updated accordingly. For this case the desired frame-rate in the Open Inventor node must be set to 4 times the user's desired frame-rate since the Open Inventor draw callback is called once for each of the viewers. Figure *5.38* shows the results of the automatic reduction applied to the CAVE. The user also has the option of manually adjusting the resolution of the model if so desired.

5.6 Conclusions

This chapter has shown the results of the majority of the research performed in the completion of the real-time visualization goal, that being the pattern vector based mesh reduction technique and the automatic resolution selection based on the multiresolution output of the reduction algorithm. A variety of models with several different modalities were tested, including models created from synthetic range images, real range images, and digital elevation maps. This illustrates the versatility of the new algorithm. The algorithm implementation requires around 2KB of memory per initial edge for the off-line processing. The reduction proceeds at a rate of 3,000 edges per second if virtual memory is not needed and 1,000 edges per second if it is. As stated earlier, there is not one optimal set of vector weights. However, in most typical cases, the geometry, boundary, and color weights are set to about the same value and the normal and curvature weights slightly lower. The curvature and normals seem to be most sensitive to weight adjustments as a result of the normalization of the vectors, thus the reason for choosing smaller weights.

With the initial mesh reduced and the results output into a multiresolution file in Open Inventor format, constant-rate interactivity is achievable. By calculating the the number of frame being drawn per second and known the current model resolution, the user's desired frame-rate is achieved by adjusting the model to the needed resolution. The user can then interact with the

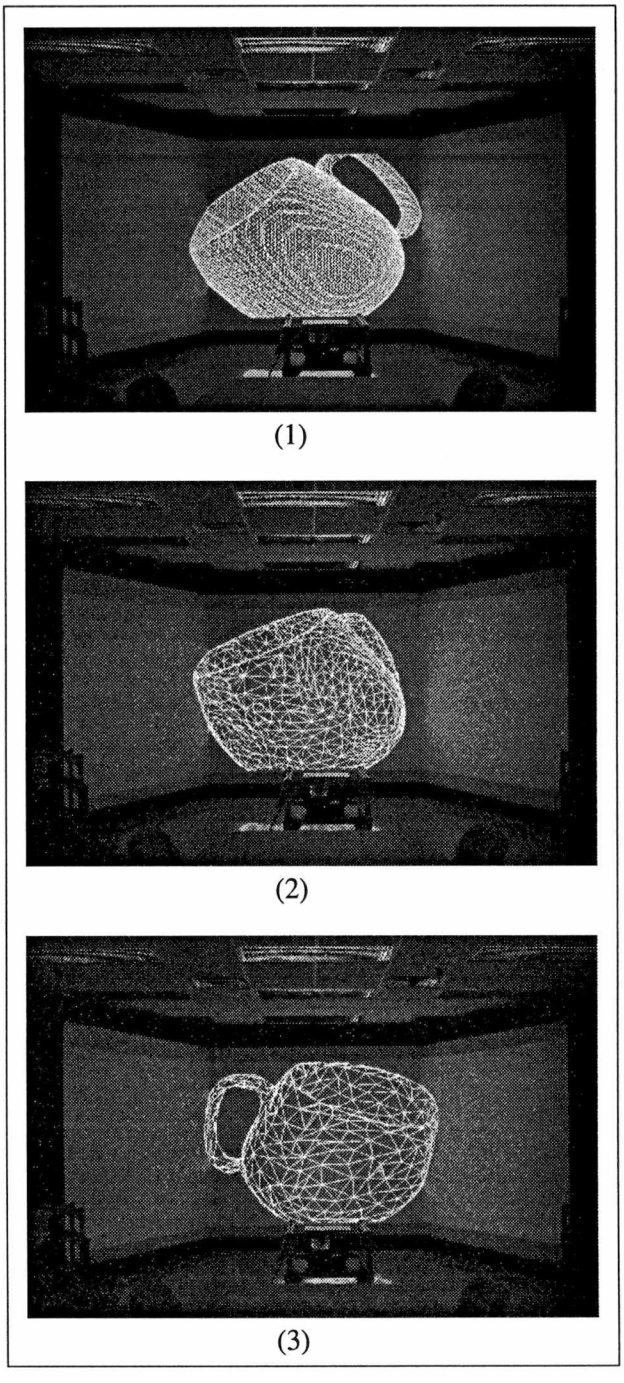


Figure 5.38: *Automatic resolution selection on performed using the CAVE display system: (1) the original model containing 38,268 triangles, (2) the resolution selected to maintain 15 frames-per-second, and (3) the resolution selected to maintain 30 frames-per-second.*

model at a lower resolution maintaining the constant update rate and view the details of the model when it is stationary. This capability is available using any Open Inventor viewer, including the one developed for the CAVE. This is the final step in the completion of the goal of visualizing multimodal 3D data in real-time by multiple users.

CHAPTER 6

Conclusions and Future Work

This research was initiated due to the recent growth in the area of photo-realistic 3D model visualization. The IRIS lab at UT has been focusing on this problem in conjunction with efforts at DOE in the area of dismantlement of old, hazardous, nuclear facilities with unknown contents. Methods are needed to quickly model these using a variety of multimodal data sets so as to effect a plan of action before sending someone into the area. The overall goal of this project at the IRIS lab deals with the creation and visualization of photo-realistic 3D models created from range and intensity data acquired from a laser range camera which can be sent into these unknown facilities to help model them. In addition to the range images, other data from the scene is also available including color, thermal, or radiation data. The IRIS overall long-term plan involves taking multiple range images from various points-of-view along with data from other sensors, combining them, and creating photo-realistic models which present the information in a useful and meaningful manner. The specific goal and focus of the research presented in this dissertation in particular is the creation of a visualization system in which these photo-realistic 3D scenes which are created can be viewed in real-time by multiple users. Specifically, the topics addressed with this research are:

- create texture-mapped models from registered pairs of range and intensity images,
- interact with the models using VR hardware,
- build a large screen, multiple user, interactive CAVE system,
- develop a method to create multiresolution models, and
- the real-time display of the models created.

Each of these were achieved, the first two being mainly implementation and the last three, actual research contributions. The multi-user portion of the research has been accomplished by designing

and building a low-cost, three projector CAVE known as the MERLIN (Multi-usER Low-cost INtegrated) visualization system. The CAVE consists of mainly off-the-shelf hardware, including three standard VGA projectors. These are mounted on adjustable height carts so they can be placed to achieve proper image edge alignment for a complete seamless image. The projectors are driven by a SGI Indigo2 which can output up to four VGA screens simultaneously. The images are projected onto a custom built screen consisting of six separate, hinged frames with one large screen stretched across them forming an image eight feet tall by twenty-seven feet wide. The seamless image is achieved through software edge matching utilizing three adjacent projective camera models.

For the real-time aspect of the project, since the hardware display rate is fixed, the number of triangles in the model must be reduced in order to increase the frame-rate to a constant user defined level. Our requirements demand the reduction method utilized satisfies the following:

- capable of handling multimodal data,
- output a multiresolution file,
- use a common data format, and
- easily extendable to incorporate new data modalities.

To meet these requirements, a new pattern vector based mesh reduction technique was created. This method handles multimodal data by assigning each component of the data to a different dimension in a feature space. This allows any form of multimodal data to be handled in an n-dimensional pattern space. Currently, these features include geometrical position, normals, color, boundary information, and curvature. It can be easily extended to include other modalities such as thermal or radiation information. Along with each pattern is an associated weight. This allows the user to determine which features influence the reduction the most though the adjustment of the pattern weights. Using the pattern space and the initial edge connection of the vertices, edge lengths can be calculated in the higher dimensional feature space. An edge collapse reduction method is then applied based on the calculated edge lengths in $I\!R^n$. By collapse one edge at a time

beginning with the smallest, a multiresolution model is built starting with the highest resolution and going to the lowest. The collapse order is stored in a dendrogram tree structure. This tree can then be traversed to recover any desired resolution of the model. This model is saved in a common data format, Open Inventor. The results of the reduction method show that good results are achievable with around 95% to 97% of the original number of triangles removed. Therefore, a model with 150,000 triangles originally can be accurately represented with only 6,000 triangles. Unfortunately, the final visual quality measure of the reduction is subjective. To account for this, the difference between projected views of original and reduced models is given.

Using the multiresolution models, a constant-rate interactivity level is achieved. The appropriate resolution is selected automatically by calculating the needed resolution based on the user-defined frame-rate and the current triangle fill rate. By setting the model to the lower resolution, the desired interactivity level is achieved while interacting with the model displayed on the CAVE or any other display device. Then when the model is stationary, the highest resolution version of the model is shown giving the details therein.

Three unique contributions have resulted from this research. The first of which is the low-cost, large screen display system for visualization. This system allows multiple users to view the data and is only a fraction of the cost of other large screen systems. Another contribution is the new pattern vector based mesh reduction technique. It is capable of handling multimodal data, outputs multiresolution files, uses a common data format, and is easily extendable to incorporate new data modalities. The final contribution is the use of a dendrogram tree structure to represent the multiresolution model. This allows for quick changes between resolutions so that a constant user interactivity rate can be achieved by means of varying the resolution.

6.1 Future Work

There are several extensions possible to this work, both general and specific to this implementation. Many of the extensions specific to this implementation involve the upgrade of hardware to increase the performance and features of the system. One of the most notable is the addition of stereo capability so the users can see the objects projected in 3D. This requires better projec-

tors which are capable of the higher refresh rates associated with stereo projection and computing hardware that is capable of producing the stereo projections. Increasing the resolution beyond 640×480 for each screen would also be desirable. One problem that occurred during the building of the CAVE system was the lack of keystone control for the projectors. This feature would greatly reduce the time needed to physically align the projectors for edge matching. Also, the color from projector to projector varies slightly. Being able to adjust the color on each projector is also a desired feature to have. These hardware upgrades increase the cost of the system, but still are relatively low. One last hardware enhancement of the system would be to remove the second monitor now at the user's workstation. The standard video output and the ICO video output from the SGI are separate and currently drives two separate monitors, but only one is available at a time. Using an automatic A/B video switch running to one monitor would allow the removal of the second monitor. Also, during the final stages of this research, the computer used to interface the dataglove was needed for another project. Re-integration into the final system is simply a matter of adding another machine with two available serial ports and running the server software.

Along with the hardware improvements, several software additions can also be made. The first of these is an extension of the constant-rate interactivity onto a PC platform. Since Open Inventor is available for the PC, this should just require a recompilation of the SoMRFaceSet node on the PC. Another software addition is the ability to integrate real-time geomorphs into an Open Inventor node. This would aid in the visualization of the models. Geomorphs reduce the popping visible to the user, but require many geomorphs for each object, one for each transition in a multiresolution model. Once the lower resolution model is determined from the automatic constant frame-rate calculation, a geomorph could be created from the low resolution model to the high resolution model. This could then be used to at the transition state between the resolutions to reduce popping.

As mentioned in the results chapter, the mesh reduction can result in small features being deleted or the model being broken. This is not a problem for our application, but some may prefer an algorithm which is topologically preserving. This could be added as an option to the pattern vector based method. The current algorithm does not label the types of faces based upon the types

of edges. The face removal is based entirely upon the edge collapse. Therefore, a face with two external edges can be removed causing the model to break in two or remove a feature which has previously been reduced to one triangle, thus changing the topology. By simply not allowing this removal in a manner similar to not allowing faces to flip during reduction, the topology will remain the same.

The vector based algorithm can also be used for other purposes. One is segmentation of the data. Since the reduction method is feature based, once the vectors are created, the data can be segmented in the vector space using standard pattern vector clustering techniques. The scenes that we are reconstructing contain many objects. Most multiresolution techniques are based upon having one object in the scene. Building models based upon the segmentation would allow each object to be represented individually and maintain the general structure of the environment if reduction is not allowed across segmentation boundaries.

Another area for future expansion for the multiresolution display is the addition of view-dependent refinement. Since the dendrogram is a binary tree, view-dependent refinement can be accomplished by moving down each branch of the tree based on a criteria other than the initial collapse/split order created from the mesh reduction technique. These criteria can be based upon such characteristics as view frustum or distance from the view-point.

One item that is missing from the area of mesh reduction, as mentioned previously, is a head-to-head comparison of different methods. This is currently not possible because no standard data format or benchmarking is available. All the algorithms are run on different models using different data formats on different machines using different error metrics. A benchmark setup is needed so each algorithm can be tested on an equal playing field.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] *Virtual Reality — State of the Art*, St. Petersburg, Russia, September 1993.

[2] J. A. Adam. Virtual Reality Is For Real. *IEEE Spectrum*, pages 22–29, October 1993.

[3] Maneesh Agrawala, Andrew Beers, Bernd Fröhlich, Pat Hanrahan, Ian McDowall, and Mark Bolas. The Two-User Responsive Workbench: Support for Collaboration Through Individual Views of a Shared Space. In *Proceedings of SIGGRAPH '97*, pages 327–332, Los Angeles, CA, August 1997.

[4] R. L. Andersson. A Real Experiment in Virtual Environments: A Virtual Batting Cage. *Presence*, 2(1):16–33, Winter 1993.

[5] Steve Aukstakalnis and David Blatner. *Silicon Mirage: The Art and Science of Virtual Reality*. Peach Pit Press, 1992.

[6] Paul J. Besl. *Surfaces in Range Image Understaning*. Springer-Verlag, New York, NY, 1988.

[7] D. K. Bhatnager. Position Trackers and Head Mounted Display Systems: A Survey. Technical report, March 1993.

[8] M. T. Bolas, I. McDowall, and R. Mead. Applications Drive VR Interface Selection. *IEEE Computer*, pages 72–75, July 1995.

[9] Louis Brill. Hotbeds of Cool Research. *IRIS Universe*, (36):60–62, Summer 1996.

[10] David J. Brown and Rebecca McClen-Novick. Jaron Lanier – Virtual Genius. *Magical Blend Magazine*, (48), 199?

[11] Samuel G. Burgiss, Ross T. Whitaker, and Mongi A. Abidi. Range image segmentation through pattern analysis of multi-scale difference information. In *Proceedings of the SPIE Conference on Intelligent Robots and Computer Vision*, Pittsburgh, PA, October 1997.

[12] Brian Carlson. 3-d visualization and virtual reality in medical apps – now. *Advanced Imaging*, pages 36–38,69, July 1996.

[13] C. H. Chen and A. C. Kak. A Robot Vision System for Recognizing 3D Objects in Low-order Polynomial Time. In *IEEE Transactions on Systems, Man, and Cybernetics*, volume 19, pages 1535–1563, 1989.

[14] M. W. Chu. Polhemus Coordinates and Polhemus-Puma Conversion. Technical report, University of Rochester, Rochester, NY, 1992.

[15] A. Ciampalini, Paolo Cignoni, Claudio Montani, and Roberto Scopigno. *Multiresolution Decimation based on Global Error*. 1997.

[16] Paolo Cignoni, Claudio Montani, Enrico Puppo, and Roberto Scopigno. Multiresolution representation and visualization of volume data. Technical Report C97-05, January 1997.

[17] James H. Clark. Hierarchical geometric models for visible surface algorithms. *CACM*, 19(10):547–554, Oct. 1976.

[18] Jonathan Cohen, Amitabh Varshney, Dinesh Manocha, Greg Turk, Hans Weber, Pankaj Agarwal, Frederick Brooks, and William Wright. Simplification envelopes. In *SIGGRAPH '96 Proc.*, pages 119–128, Aug. 1996.

[19] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti. Surround-Screen Projection-Based Virtual Reality: The Design and Implementaion of the CAVE. In *Computer Graphics (Proceedings of SIGGRAPH '93)*, pages 135–142, August 1993.

[20] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart. The CAVE: Audio Visual Experience Automatic Virtual Environment. *Communications of the ACM*, 35(6):65–72, June 1992.

[21] T. A. DeFanti, D. J. Sandin, and C. Cruz-Neira. A 'Room' With a 'View'. *IEEE Spectrum*, pages 30–33, October 1993.

[22] Tony DeRose, Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Fitting of surfaces to scattered data. In *Proceedings of SPIE Conference on Surfaces in Computer Vision and Graphics*, volume 1830, pages 212–220, Boston, MA, November 1992.

[23] Tony DeRose, M. Lounsbery, and J. Warren. Multiresolution analysis for surface of arbitrary topological type. Technical report, University of Washington, Seattle, WA, 1993.

[24] Sir Launcelot du Lake. CyberMan from Logitech. *GameBytes*, 21.

[25] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. In *SIGGRAPH '95 Proc.*, pages 173–182. ACM, Aug. 1995.

[26] Grant Ellis. They're Not Making 'Em Like They Used To. *IRIS Universe*, (36):28–32, Summer 1996.

[27] David L. Elsner, Ross T. Whitaker, and Mongi A. Abidi. 3d model creation through volumetric fusion of multiple range images. In *Proceedings of the SPIE Conference on Decentralized Control in Autonomous Robotic Systems*, Pittsburgh, PA, October 1997.

[28] Mark D. Elstrom, Philip W. Smith, and Mongi A. Abidi. Stereo-based registration of ladar and color imagery. In *Proceedings of the SPIE Conference on Intelligent Robots and Computer Vision*, Boston, MA, November 1998.

[29] Francine Evans, Steven Skiena, and Amitabh Varshney. Optimizing triangle strips for fast rendering. In *IEEE Visualization 96 Proceedings*, pages 319–326. IEEE, October 1996.

[30] J. D. Foley, A. A. van Dam, S. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley Publishing Company, Reading, MS, 1990.

[31] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee. *Robotic Control, Sensing, Vision, and Intelligence*. McGraw-Hill, 1987.

[32] Thomas A. Funkhouser. *Database and Display Algorithms for Interactive Visualization of Architectural Models*. PhD thesis, CS Division, UC Berkeley, 1993.

[33] Michael Garland and Paul S. Heckbert. Fast polygonal approximation of terrains and height fields. submitted for publication.

[34] Michael Garland and Paul S. Heckbert. Fast polygonal approximation of terrains and height fields. Technical report, CS Dept., Carnegie Mellon U., Sept. 1995.

[35] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH '97 Proc.*, pages 209–216. ACM, 1997.

[36] J. C. Goble, K. Hinckley, R. Pausch, J. W. Snell, and N. F. Kassell. Two-Handed Spatial Interface Tools for Neurosurgical Planning. *IEEE Computer*, pages 20–26, July 1995.

[37] R. C. Gonzalez and P. Wintz. *Digital Image Processing*. Addison-Wesley Publishing Company, Reading, MS, 1987.

[38] Christopher S. Gourley and Mongi A. Abidi. Virtual reality hardware for use in interactive 3d data fusion and visualization. In *Proceedings of the SPIE Conference on Decentralized Control in Autonomous Robotic Systems*, Pittsburgh, PA, October 1997.

[39] Paul Haeberli and Mark Segal. Texture mapping as a fundamental drawing primitive. *SGI White Paper*, June 1993.

[40] D. Hancock. Prototyping the Hubble Fix. *IEEE Spectrum*, pages 34–39, October 1993.

[41] Taosong He, L. Hong, A. Kaufman, A. Varshney, and S. Wang. Voxel-based object simplification. In *Proc. Visualization '95*. IEEE Comput. Soc. Press, 1995.

[42] Taosong He, L. Hong, A. Kaufman, A. Varshney, and S. Wang. Controlled topology simplification. *IEEE Transactions on Visualization and Computer Graphics*, 2(2):171–184, 1996.

[43] Paul S. Heckbert and Michael Garland. Multiresolution modeling for fast rendering. In *Proc. Graphics Interface '94*, pages 43–50, Banff, Canada, May 1994. Canadian Inf. Proc. Soc.

[44] Paul S. Heckbert and Michael Garland. Survey of polygonal surface simplification algorithms. In *SIGGRAPH '97 Course Notes CD-ROM, Course 25: Multiresolution Surface Modeling*. ACM SIGGRAPH, August 1997.

[45] Michael Heim. *The Metaphysics of Virtual Reality*. Oxford Unversity Press, New York, 1993.

[46] Paul Hinker and Charles Hansen. Geometric optimization. In *Proc. Visualization '93*, pages 189–195, San Jose, CA, October 1993.

[47] L. F. Hodges, R. Kooper, T. C. Meyer, B. O. Rothbaum, D. Opdyke, J. J. de Graaff, J. S. Williford, and M. M. North. Virtual Environments for Treating the Fear of Heights. *IEEE Computer*, pages 27–34, July 1995.

[48] Hugues Hoppe. *Surface Reconstruction from Unorganized Points*. PhD thesis, Dept. of Computer Science and Engineering, U. of Washington, 1994.

[49] Hugues Hoppe. Progressive meshes. In *SIGGRAPH '96 Proc.*, pages 99–108, Aug. 1996.

[50] Hugues Hoppe. View-dependent refinement of progressive meshes. In *Proceedings of SIGGRAPH '97*, pages 189–198, Los Angeles, CA, August 1997.

[51] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 71–78, July 1992.

[52] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In *SIGGRAPH '93 Proc.*, pages 19–26, Aug. 1993.

[53] Linda Jacobson. Visiting the Virtually Real. *IRIS Universe*, (36):18–19, Summer 1996.

[54] Andy Johnson, Jason Leigh, and Jim Costigan. Multiway Tele-Immersion at Supercomputing '97. *IEEE Transactions On Computer Graphics and Applications*, July 1998.

[55] Alan D. Kalvin, Court B. Cutting, B. Haddad, and M. E. Noz. Constructing topologically connected surfaces for the comprehensive analysis of 3D medical structures. In *Medical Imaging V: Image Processing*, volume 1445, pages 247–258. SPIE, Feb. 1991.

[56] Alan D. Kalvin and Russel H. Taylor. Superfaces:polygonal mesh simplification with bounded error. *IEEE Computer Graphics and Appl.*, 16(3), May 1996.

[57] G. Drew Kessler, Larry F. Hodges, and Neff Walker. Evaluation of the CyberGlove as a Whole Hand Input Device. *Transactions On Human Computer Interactions*, December 1995.

[58] Myron W. Krueger. *Artificial Reality*. Addison-Wesley, Reading, MA, 1983.

[59] Myron W. Krueger. *Artificial Reality II*. Addison-Wesley, Reading, MA, 1991.

[60] W. Krüger, C. A. Bohn, B. Froöhlich, H. Schüth, W. Strauss, and G. Weche. The Responsive Workbench: A Virtual Work Environment. *IEEE Computer*, pages 42–48, July 1995.

[61] J. Kuhl, D. Evans, Y. Papelis, R. Romano, and G. Watson. The Iowa Driving Simulator: An Immersive Research Environment. *IEEE Computer*, pages 35–41, July 1995.

[62] Jaron Lanier. Virtual reality: The promise of the future. *Interactive Learning International*, 8(4):275–279, October-December 1992.

[63] D. Lau. Investigation and Application of Virtual Reality Technology to Low Cost Graphics Workstations. Technical report, June 1994.

[64] Eric D. Lester, Ross T. Whitaker, and Mongi A. Abidi. Feature extraction, image segmentation, and scene reconstruction. In *Proceedings of the SPIE Conference on Decentralized Control in Autonomous Robotic Systems*, Pittsburgh, PA, October 1997.

[65] David Luebke. Hierarchical structures for dynamic polygonal simplification. Technical report, University of North Carolina at Chapel Hill, Department of Computer Science, 1996.

[66] David Luebke and Carl Erikson. View-dependent simplification of arbitrary polygonal environments. In *SIGGRAPH '97 Proc.* ACM, 1997.

[67] Jonathan Luskin. The Universe in a Cave. *IRIS Universe*, (35):28–32, Spring 1996.

[68] T. W. Mastaglio and R. Callahan. A Large-Scale Complex Virtual Environment for Team Training. *IEEE Computer*, pages 49–56, July 1995.

[69] Charmaine Moyer and Wendy Ferguson. *Indigo^2 IMPACT Channel Option Installation Guide.* Silicon Graphics, Inc., Mountain View, CA, 1996.

[70] National Research Council. *Virtual Reality Scientific and Technological Challenges.* National Academy Press, Washington, DC, 1995.

[71] Jackie Neider, Tom Davis, and Mason Woo. *OpenGL Programming Guide.* Addison-Wesley Publishing Company, Reading, MS, 1993.

[72] J. G. Neugebauer, *et al.* Applications of Virtual Reality in Endoscopy. Technical report, Fraunhofer-Institute for Manufacturing Engineering and Automation (IPA), Stuttgart, Germany.

[73] Polhemus. *3Space® Fastrak® User's Manual.* Colchester, VT, November 1993.

[74] M. F. Polis, S. J. Gifford, and D. M. McKeown Jr. Automating the Construction of Large-Scale Virtual Worlds. *IEEE Computer*, pages 57–65, July 1995.

[75] Jovan Popović and Hugues Hoppe. Progressive simplical complexes. In *Proceedings of SIGGRAPH '97*, pages 217–224, August 1997.

[76] D. R. Pratt, M. Zyda, and K. Kelleher. Virtual Reality: In the Mind of the Beholder. *IEEE Computer*, pages 17–19, July 1995.

[77] Kari Pulli, Tom Duchamp, Hugues Hoppe, John McDonald, Linda Shapiro, and Werner Stuetzle. Robust meshes from multiple range maps. In *Inernational Conference on Recent Advances in 3-D Digital Imaging and Modeling Proc.*, pages 205–211. IEEE, May 1997.

[78] Enrico Puppo and Roberto Scopigno. Simplificatio, LOD, and multiresolution - principles and applications. In *Eurographics '97 Tutorial Notes, Eurographics Associations*, Aire-la-Ville, France, 1997.

[79] Cindy Reed. Understanding how textures work. Technical report.

[80] Kevin J. Renze and James H. Oliver. Generalized surface and volume decimation for unstructured tessellated domains. In *VRAIS '96 (IEEE Virtual Reality Annual Intl. Symp.)*, Mar. 1996. submitted.

[81] Report of an NSF Invitational Workshop. Research Directions in Virtual Environments. Technical report, University of North Carolina, Chapel Hill.

[82] B. Roehl. The Logitech Cyberman. Technical report, 1994.

[83] B. Roehl. The Fifth Dimension Glove/ The Virtual I/O i-glasses! HMD. Technical report, 1995.

[84] Jarek Rossignac. Geometric simplification and compression. In *SIGGRAPH '97 Course Notes CD-ROM, Course 25: Multiresolution Surface Modeling.* ACM SIGGRAPH, August 1997.

[85] Jarek Rossignac and Paul Borrel. Multi-resolution 3D approximations for rendering complex scenes. Technical report, Yorktown Heights, NY 10598, Feb. 1992. IBM Research Report RC 17697. Also appeared in *Modeling in Computer Graphics*, Springer, 1993.

[86] Jarek Rossignac and Paul Borrel. Multi-resolution 3D approximations for rendering complex scenes. In B. Falcidieno and T. Kunii, editors, *Modeling in Computer Graphics: Methods and Applications*, pages 455–465, Berlin, 1993. Springer-Verlag. Proc. of Conf., Genoa, Italy, June 1993. (Also available as IBM Research Report RC 17697, Feb. 1992, Yorktown Heights, NY 10598).

[87] K. Russel, *et al.* Unencumbered Virtual Environments. Technical report, Perceptual Computing Section, Media Laboratory, MIT, Cambridge, MA.

[88] Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit, An Object-Oriented Approach To 3D Graphics*. Prentice Hall, 1996.

[89] William Schroeder. A topology modifying progressive decimation algorithm. In *SIGGRAPH '97 Course Notes CD-ROM, Course 25: Multiresolution Surface Modeling*. ACM SIGGRAPH, August 1997.

[90] William Schroeder and Tom Citriniti. Decimating polygon meshes. *Dr. Dobb's Journal*, July 1997.

[91] William J. Schroeder and Boris Yamrom. A compact cell structure for scientific visualization. In *SIGGRAPH '94 Course Notes CD-ROM, Course 4: Advanced Techniques for Scientific Visualization*, pages 53–59. ACM SIGGRAPH, July 1994.

[92] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. *Computer Graphics (SIGGRAPH '92 Proc.)*, 26(2):65–70, July 1992.

[93] D. Song and M. L. Norman. Cosmic Explorer: A Virtual Reality Environment for Exploring Cosmic Data. Technical report, National Center for Supercomputing Applications, Champaign, IL, March 1993.

[94] Marc Soucy. Innovmetric's multiresolution modeling algorithms. In *SIGGRAPH '97 Course Notes CD-ROM, Course 25: Multiresolution Surface Modeling*. ACM SIGGRAPH, August 1997.

[95] Marc Soucy, Alain Croteau, and Denis Laurendeau. A multi-resolution surface model for compact representation of range images. In *Intl. Conf. on Robotics and Automation*, pages 1701–1706, May 1992.

[96] Marc Soucy and Denis Laurendeau. Surface modeling from dynamic integration of multiple range views. In *Proc. 11th Intl. Conf. on Pattern Recognition*, pages 449–452, 1992.

[97] Edwin H. Spanier. *Algebraic Topology*. Springer-Verlag, New York, 1981.

[98] L. Stark, *et al.* Telerobotics: Display, Control, and Communication Problems. *IEEE Journal of Robotics and Automation*, RA-3(1):67–75, February 1987.

[99] W. M. Strommer, *et al.* Transputer-based Virtual Reality Workstation as Implemented for the Example of Industrial Robot Control. Technical report, Fraunhofer-Institute for Manufacturing Engineering and Automation (IPA), Stuttgart, Germany.

[100] Ivan E. Sutherland. The Ultimate Display. In *Proceedings of the IFIPS Congress*, volume 2, pages 506–508, 1965.

[101] Ivan E. Sutherland. A Head-mounted Three-dimensional Display. In *1968 Fall Joint Computer Conference, AFIPS Conference Proceedings*, volume 33, pages 757–764, 1968.

[102] J. T. Tou and R. C. Gonzalez. *Pattern Recognition Principles.* Addison-Wesley Publishing Company, Reading, MS, 1974.

[103] J. E. Townsend. *PowerGlove FAQ.* J. E. Townsend Compilation, 1993.

[104] L. H. Tsoukalas and R. E. Uhrig. *Fuzzy and Neural Approaches in Engineering.* Under Preparation, 1994.

[105] C. P. Tung and A. C. Kak. Automatic Learning of Assembly Tasks Using a DataGlove System. In *Proceedings of the 1995 IEEE/RSJ International Confernce on Intelligent Robots and Systems*, pages 1–8, July 1995.

[106] Greg Turk. Re-tiling polygonal surfaces. *Computer Graphics (SIGGRAPH '92 Proc.)*, 26(2):55–64, July 1992.

[107] Greg Turk and Marc Levoy. Zippered Polygon Meshes from Range Images. In *Proceedings of SIGGRAPH '94*, pages 311–318, Orlando, FL, July 1994.

[108] Amitabh Varshney. *Hierarchical Geometric Approximations.* PhD thesis, Dept. of CS, U. of North Carolina, Chapel Hill, 1994. TR-050.

[109] Amitabh Varshney. A hierarchy of techniques for simplifying polygonal models. In *SIGGRAPH '97 Course Notes CD-ROM, Course 25: Multiresolution Surface Modeling.* ACM SIGGRAPH, August 1997.

[110] Amitabh Varshney, Pankaj K. Agarwal, Frederick P. Brooks, Jr., William V. Wright, and Hans Weber. Generating levels of detail for large-scale polygonal models. Technical report, Dept. of CS, Duke U., Aug. 1995.

[111] Virtual Technologies. Cyber*Glove*^{TM} *User's Manual.* Palo Alto, CA, June 1993.

[112] H. J. Warnecke, *et al.* Virtual Reality for Improved Human-Computer Interaction in Robotics and Medicine. Technical report, Fraunhofer-Institute for Manufacturing Engineering and Automation (IPA), Stuttgart, Germany.

[113] P. Wegner. Concepts and Paradigms of Object-Oriented Programming. June 1990.

[114] Josie Wernecke. *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor.* Addison-Wesley Publishing Company, Reading, MS, 1994.

[115] Josie Wernecke. *The Inventor Toolmaker.* Addison-Wesley Publishing Company, Reading, MS, 1994.

[116] Lance Williams. Pyramidal parametrics. *Computer Graphics (SIGGRAPH '83 Proc.)*, 17(3):1–11, July 1983.

[117] Laurana M. Wong, Christophe Dumont, and Mongi A. Abidi. An algorithm for finding the next best view in object reconstruction. In *Proceedings of the SPIE Conference on Intelligent Systems and Advanced Manufacturing*, Boston, MA, November 1998.

[118] Julie C. Xia, Jihad El-Sana, and A. Varshney. Adaptive real-time level-of-detail-based rendering for polygonal models. *IEEE Transactions on Visualization and Computer Graphics*, June 1997.

[119] T. Yoshikawa. *Foundations of Robotics: Analysis and Control.* MIT Press, 1990.

[120] Denis Zorin, Peter Schröder, and Wim Sweldens. Interactive multiresolution mesh editing. In *SIGGRAPH '97 Proc.* ACM, 1997.

APPENDICES

APPENDIX A

Theory and Background

This appendix presents the background information and theory that is needed for implementation of the various algorithms and methods used for visualization. At the lowest levels, visualization relies heavily on mathematical geometric transforms. Also, several computer graphics techniques are used to aid in the presentation of the models created. Other tools, such as neural networks and range scanning are also discussed and their relation to the research presented here.

A.1 Basic 3D Transforms

We are using a data set returned from the range scanner or simulator in the form of a range image. Each point in the set is a distance from a point along a ray from the scanner. These must be converted into 3D data points for visualization. Each point will have a three-dimensional coordinate (x,y,z) along with a color value, either grey-scale intensity or RGB. With this in mind, we need to establish some basic 3D transforms both for manipulating and rearranging the data as well as for display and visualization of the data. We will be using a right-handed coordinate system. Some computer graphics and image processing books use a left-handed system so that the positive z values will be away from the viewer which seems more natural. However, the software used for implementation (OpenGL/Open Inventor) inherently uses a right-handed system, therefore that is what we use. This is also the standard mathematical convention. First of all, each point will be represented as an augmented vector:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{A.1}$$

The multiple data sets that we have will need to be translated and rotated in order for them to align. The data can be transformed using a 4×4 matrix. This is referred to as a modeling transform in

OpenGL. The first transform matrix is a simple translation:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.2}$$

This will move the point from (x,y,z) to $(x + x_0, y + y_0, z + z_0)$. Next are rotations about the various axes by an angle θ. Rotation about the x, y, and z axes are given as follows:

$$\mathbf{R_x} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.3}$$

$$\mathbf{R_y} = \begin{bmatrix} cos\theta & 0 & sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -sin\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.4}$$

$$\mathbf{R_z} = \begin{bmatrix} cos\theta & -sin\theta & 0 & 0 \\ sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.5}$$

Scaling by factors S_x, S_y, and S_z along the x, y, and z axes is given by the matrix:

$$\mathbf{S} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.6}$$

Another useful transform is the perspective transform. This is used to create a pin hole perspective camera model which is used for display of the data. The perspective transform is given by:

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{\lambda} & 1 \end{bmatrix} \tag{A.7}$$

where λ is the focal length of the lens in the camera model as shown in Figure *A.1*. Also useful is the inverse transform matrix. For the transform matrices used here, this is calculated as follows:

$${}^{B}\mathbf{T}_{A} = {}^{A}\mathbf{T}_{B}{}^{-1} = \left[\begin{array}{c|c} {}^{A}\mathbf{R}_{B}{}^{T} & -{}^{A}\mathbf{R}_{B}{}^{T}\,{}^{A}\mathbf{p}_{B} \\ \hline \mathbf{0} & 1 \end{array}\right] \tag{A.8}$$

where ${}^{A}\mathbf{T}_{B}$ is the transform of frame B relative to frame A and ${}^{B}\mathbf{T}_{A}$ is the transform of frame A relative to frame B. So for a point in frame B to be represented in frame A would require the point to be multiplied by ${}^{A}\mathbf{T}_{B}$. This notation is that used by Yoshikawa [119].

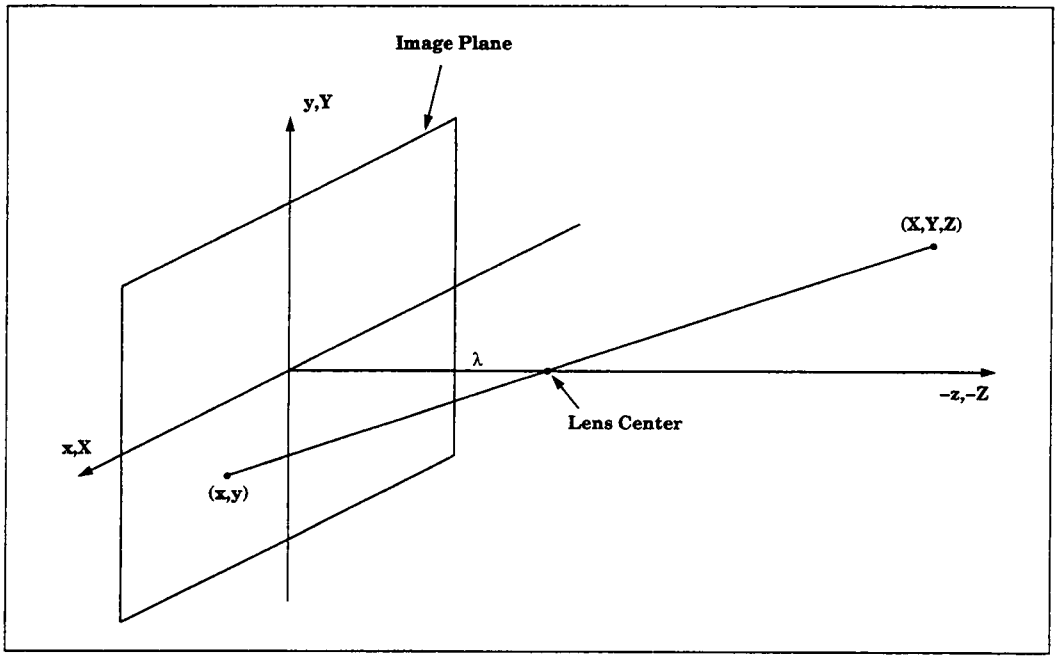


Figure A.1: *Pinhole camera model with a focal length of λ for a right-handed coordinate system.*

For implementation the Open Inventor 3D toolkit is used. Each of the transforms mentioned is represented as an Open Inventor node. The data points are stored in a *Coordinate3* node. *Translation*, *Rotation*, and *Scale* nodes are also part of the Open Inventor package. The rotation nodes include rotation about a single axis (*RotateXYZ*) or rotation about an arbitrary vector (*Rotation*). A *Transform* node is available, but this node consists of a translation followed by general rotation and scale. A general transform can be implemented as a *MaxtrixTransform* node.

A.2 Camera Model

For visualization of an object, a camera model is needed to properly view the scene. In most cases, a simple pin hole camera model (see Figure *A.1*) is used which is implemented by performing a perspective transform [31, 37, 30]. The formation of the complete camera model involves the calculation of the matrix containing all translation, rotation, and perspective information from the cameras position to that of a reference frame. Gonzalez [31] gives a detailed example of a pinhole camera model, including gimbal offsets and many other specifics. For Open Inventor, the

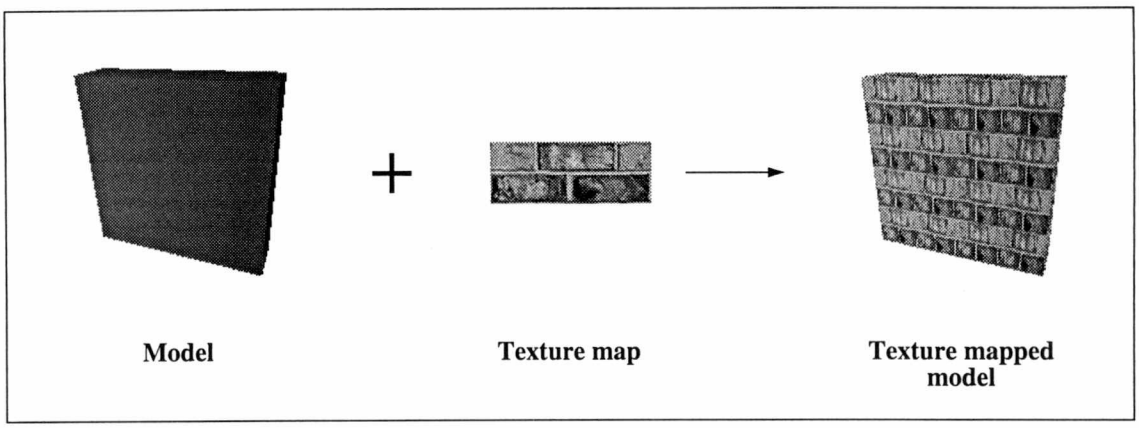


Figure A.2: *Texture of a brick mapped onto a cube.*

camera model is implemented as a *Viewer*. The Open Inventor viewers can have a perspective or orthogonal camera model to map the data. Here, a perspective model is used. The vertical field-of-view of the camera model is set as the height angle of the viewer and the horizontal as an aspect ratio with respect to the vertical angle.

A.3 Texture Mapping

We wish to add realism to the models that are generated and also to merge the intensity and range data. To accomplish this, the intensity will be texture mapped onto the model created from the range data. Basic texture mapping paints an image (texture) onto a polygon rendered in a scene by assigning texture coordinates to the vertices of the polygon resulting in portions of the texture being mapped onto the polygon [30, 79, 114, 39]. It can be thought of as placing a decal or shrink-wrap around the surface as shown in Figure *A.2*. This is a powerful method for adding realism to a scene. Because this method is so useful, it is becoming a standard method for rendering in graphics hardware and software. Hardware texture mapping makes this technique available with little or no overhead on scene update rates.

Textures can be two or three dimensional. The textures used here are two dimensional (intensity images returned from a laser range camera). The texture space is measured in (u, v) coordinates, with u being the horizontal component, and v the vertical. The origin is the lower-left

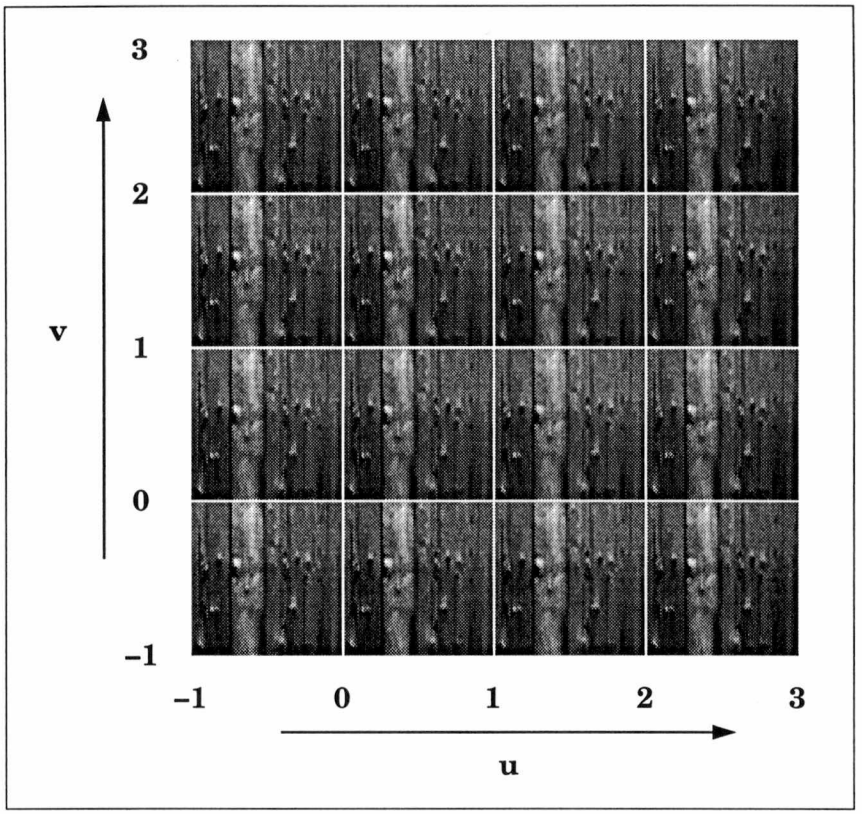


Figure A.3: *Texture coordinates shown using a repeating texture of wood paneling.*

corner with coordinates $(0, 0)$ and the upper-right corner has coordinates $(1, 1)$. These coordinates are always true for any size texture. This allows for texture repetitions when coordinates greater than 1.0 are used as shown in Figure *A.3*.

When mapping an image onto a mesh the color at each pixel of the polygon is modified by a color in the image. The image must be warped to match the mapping onto the mesh and filtered to remove components that would cause aliasing. The image is then resampled to get the final color for the textured pixel. There are different methods for calculating the new color for the pixel. Modulation multiples the pixel color by the texture color, decaling replaces the pixel color with the texture color, and blending uses the texture intensity to blend the pixel color with a constant blend color. We use modulation in order to maintain the use of the surface normals calculated for the mesh. When texture coordinates are assigned to a polygon's vertices, the texture image is interpolated across the polygon to determine texture values at each of the pixels in the

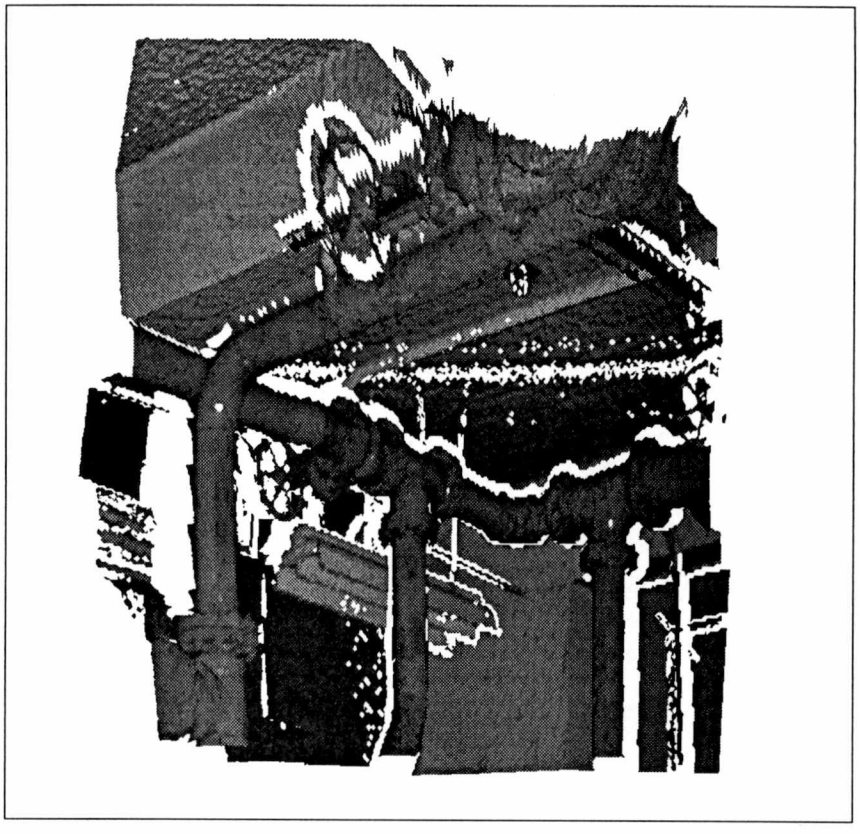


Figure A.4: *Registered intensity image texture mapped onto a triangle mesh created from a range image.*

polygon, not just at the vertices which occurs when using materials. Open Inventor handles the interpolation steps automatically. 2D textures are perfectly flat, but the surfaces they map onto are not. Therefore, an intermediate coordinate space is required. This space is defined by the coordinates (s, t), s being the horizontal component and t the vertical. The (s, t) coordinate represents the warping map used to relate the (u, v) texture image coordinates to the (x, y, z) polygon vertex coordinates as shown in Figure *A.4* [71, 30]. To aid in the visualization of the range data, we will texture map the registered intensity data onto the 3D model created from the range data to create a photo-realistic model.

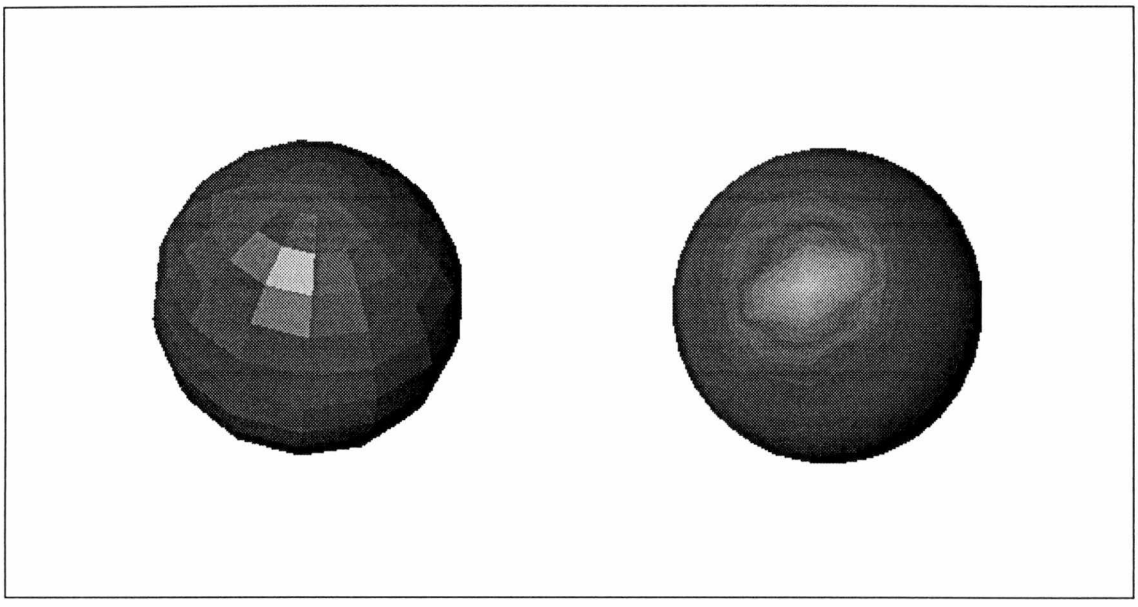


Figure A.5: *Faceted surface shown on left versus a smoothed surface on right created using true point normals.*

A.4 Calculating Surface Normals for Improved Visual Quality

The normals of the surfaces of a model are used to create the proper shading based on the lighting model. In a default model, the normal values are all the same. This causes the model to appear faceted. The true normal at each vertex is needed to give the model a smooth look. The normal is the vector that is perpendicular to the plane of the surface. A curved surface is approximated by a large number of small polygons, usually triangles. If the normals are calculated on a per surface basis, the surface will then appear faceted, or discontinuous, across the surface. If the true surface normal can be calculated at each point instead of for each surface, the rendering quality is greatly improved (see Figure *A.5*). The easiest way to accomplish this is to calculate the normal for each of the polygon surfaces and then use the average of the normals for each facet neighboring a vertex for the actual normal at that vertex. For example, the normal at the point $p2$ in Figure *A.6* would be the result of averaging the normals for each of the neighboring facets, $n1$, $n2$, $n3$, and $n4$.

To calculate normals for the vertices of the meshes we are generating, we first calculate the

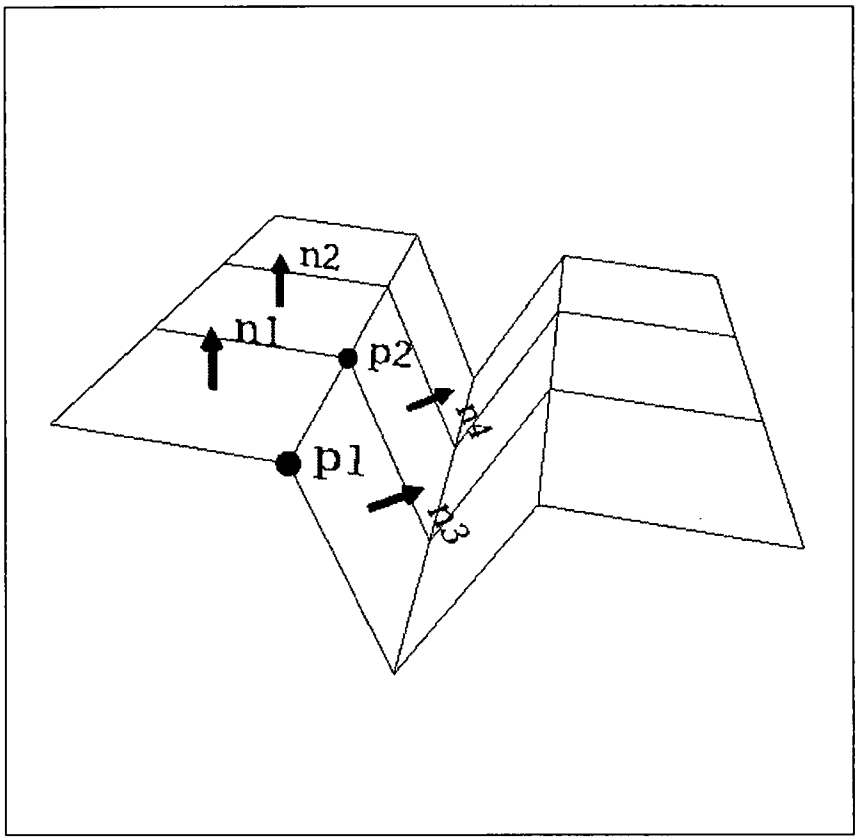


Figure A.6: *Normal calculations for points based on the normals from faceted polygon data.*

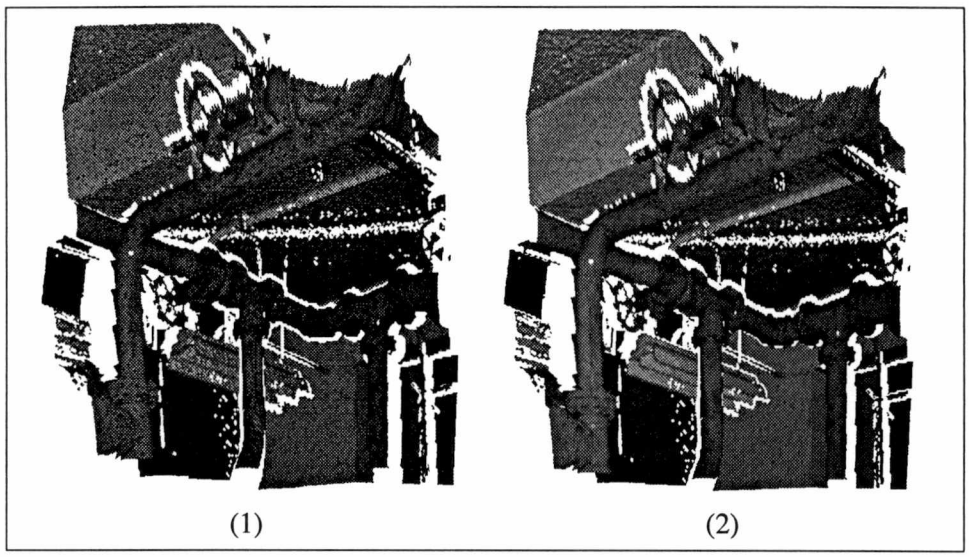


Figure A.7: *Textured polygons shown (1) with default normals and (2) with calculated true normals.*

normals for each facet as it is created. This is done by taking the cross product of the two vectors formed from the three vertices of each triangle. The normals and the vertices used to create them are stored in a data class. After all the triangle meshes have been created, then the normals for each vertex can be calculated. For each vertex, the data class is searched to find all the surfaces that include it. Then the normal for the surface can be calculated by averaging all the surface normals found. The resulting triangle mesh appears much smoother than one relying on the default faceted normals as can be seen in Figure *A.7.*

A.5 Neural Networks

This section gives a brief general overview of the use and composition of an artificial neural network. Details of the training methods and theory of the artificial neural network can be found in [104]. We are using a neural network to recognize gestures from the 23 output signals from the dataglove. These signals are input into a single layer feed-forward network and the gesture is output.

An artificial neural network (ANN) is a fault-tolerant, distributed, associative memory capable of pattern recognition which learns by example [104]. They have a biological basis and attempt

to emulate the brain's ability to learn. The neural network is a computer representation of the operation of the human brain. It attempts to mimic the brains ability to learn. It's lowest level component is the neuron, based on the brains own simple processing elements. A neural network can only perform two functions:

1. Learning
2. Recall

Artificial neural networks are also known as connectionist systems. The network is made up of interconnected neurons. Artificial neurons, also known as processing elements (PE's), neurodes, nodes or neurons, sum their weighted inputs and pass this value through a transfer function to obtain the output. The network must first be trained by example. After training, recall is almost instantaneous.

A.5.1 The Artificial Neuron

Each neuron (shown in Figure *A.8*) has several input signals and a weight associated with each signal. At the neuron, the signals are multiplied by there respective weights and summed together. A bias can also be applied to the neuron.

$$I_j = \sum_i W_{ij} X_i + bias \tag{A.9}$$

The second part of the neuron is a transfer function. The value of the weighted sum is used to generate an output value from the node via the transfer function.

$$Y_j = \Phi(I_j) \tag{A.10}$$

Commonly, this function is continuous and varies between two asymptotic values. Most commonly used are the

- sigmoidal – $\frac{1}{1+e^{-\alpha x}}$,
- signum – $+1\ for\ x > 0,\ -1\ for\ x \leq 0$, and
- atan functions.

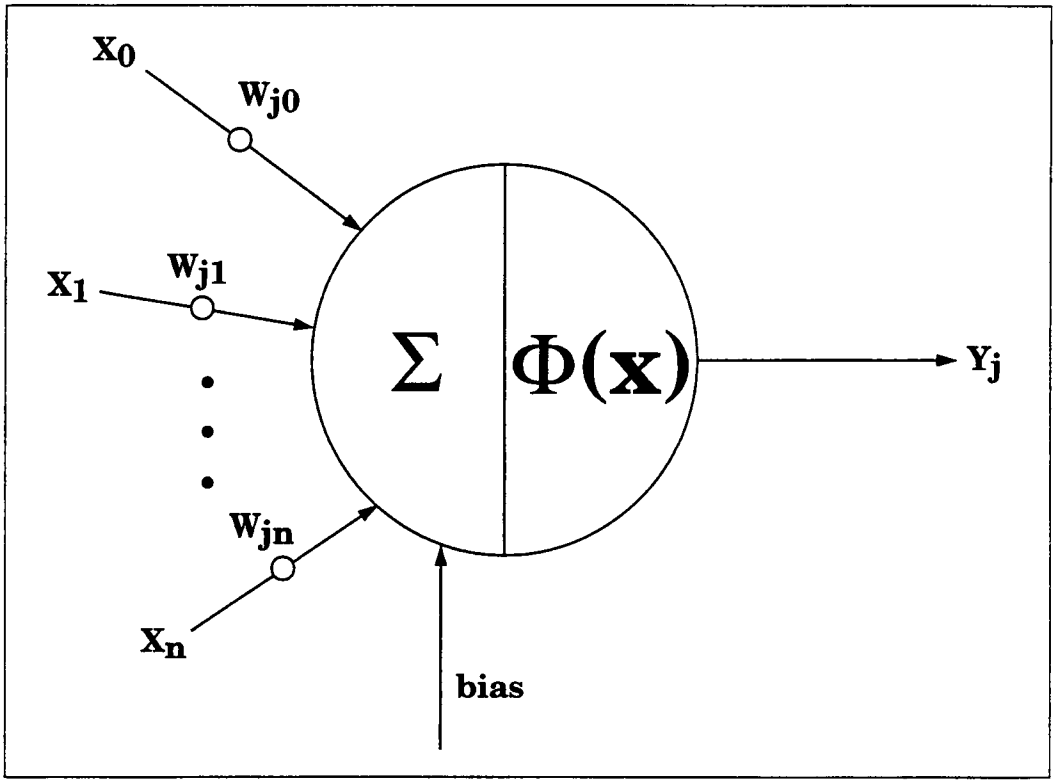


Figure A.8: *Artificial neuron.*

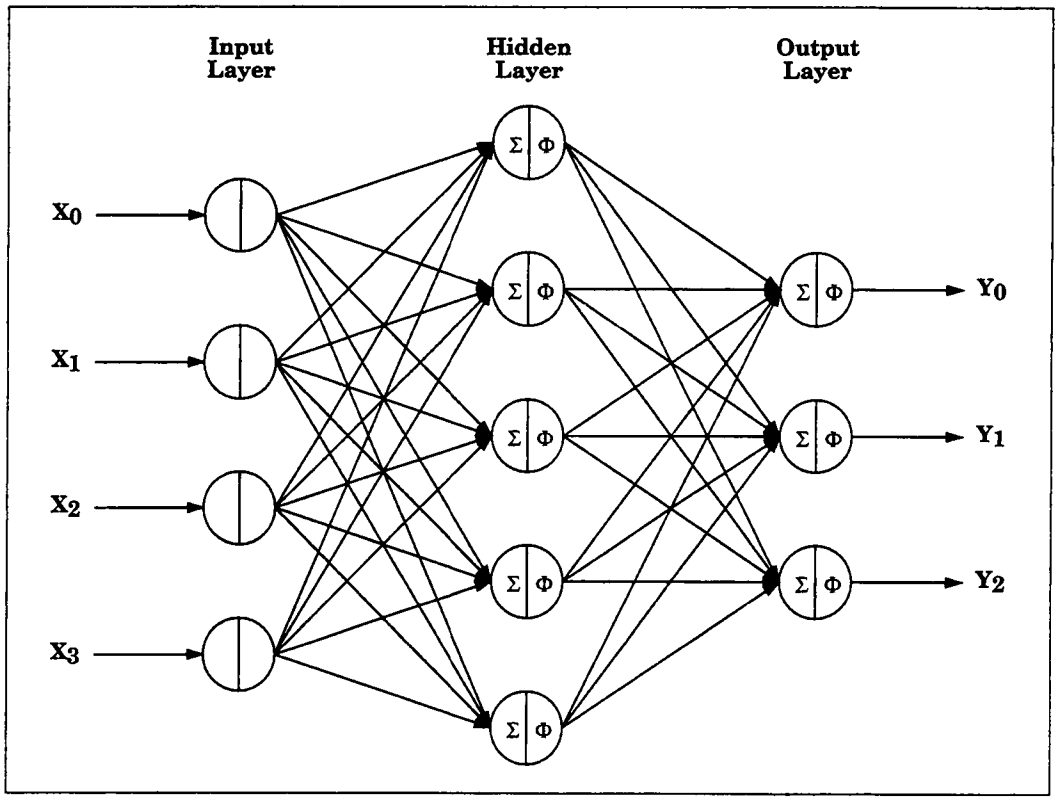


Figure A.9: *A fully connected feedforward artificial neural network with 4 inputs, 5 hidden nodes, and 3 outputs.*

A.5.2 Artificial neural network

An artificial neural network is defined as:

> a data processing system consisting of a large number of simple, highly-interconnected processing elements in an architecture inspired by the structure of the cerebral cortex of the brain [104].

In the network the PE's are usually arranged in layers with full or random connections between the layers (see Figure *A.9*). Fully connected means that each node in one layer has a weighted connection to every node in the next layer. Random connections can be viewed as fully connected layers with some weights of 0. Input nodes have no weights or transfer functions associates with them. They act as buffers to pass the data to the internal nodes. Middle layers are also referred to as hidden layers.

These networks are simple feedforward networks. Neural networks can also contain feedback connections laterally in the same layer or from the output back to the input of the network. About 80% of the neural network applications today utilize feedforward neural networks.

A.5.3 Learning

The process of adjusting the weighs on the connections until the desired results are output from the network is known as learning. There are several types of learning, including supervised, graded, and unsupervised. Backpropagation training is a gradient descent method which tries to minimize the mean squared error of the output. It proceeds by allowing the input to pass forward through the network and determines the error of the output. The change in the weights are then assumed to be proportional to the rate of change of the square error with respect to that weight and passed backward thought the network. This process continues until the output error is below a preset criteria or a maximum number of iterations is reached.

A.5.4 Recall

The process of receiving an input signal to the network and outputting a response for a given set of weights and biases is known as recall. Recall is used both during training and in the application of the network. For a feedforward neural network, after learning has occurred recall is very fast.

A.6 Range Scanning

We are creating models from a laser range scanner. In order to build the models, the manner in which the scanner captures the images and stores them needs to be understood. A laser range scanner is an orthogonal-axis angle range sensor, that is a sensor that measures the distance to a point in the environment as a function of horizontal and vertical scan angles [6]. Points are measured from a single point by casting a ray from the starting point and pivoting about the x axis by an angle ϕ and the y axis by an angle θ as shown in Figure *A.10*. This returns a distance r to the closest object along the current ray given by θ_0 and ϕ_0. The relationship between the actual

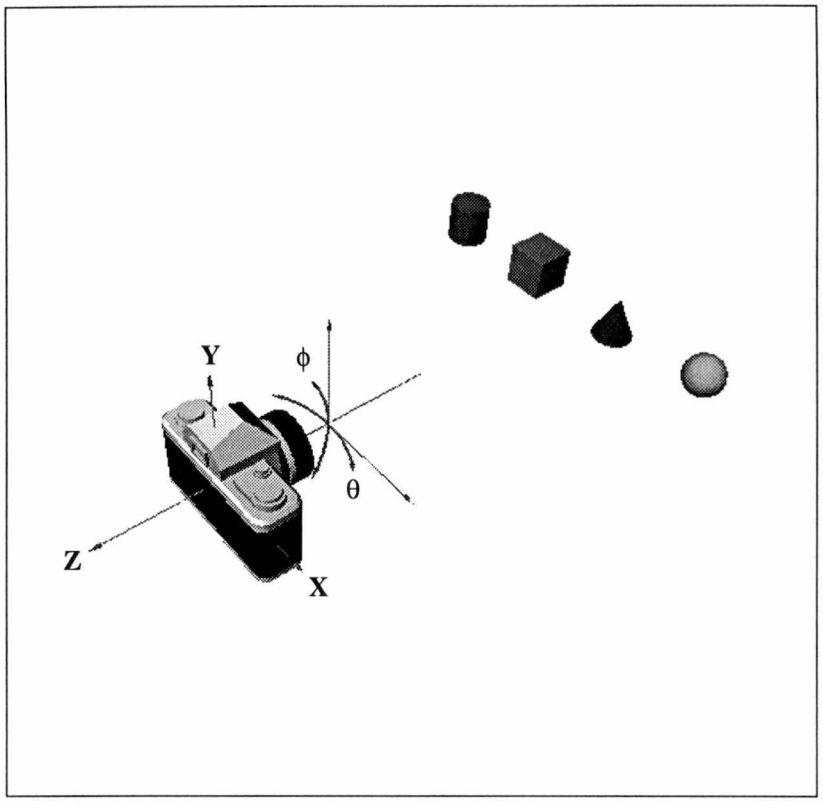


Figure A.10: *An orthogonal-axis scanner which casts a ray to the nearest object from its starting point while pivoting about its x and y axes.*

cartesian coordinates (x, y, z) and the returned distance r is given by

$$r^2 = x^2 + y^2 + z^2. \tag{A.11}$$

From the geometry of the scene the following can also be calculated:

$$x = z + \tan(\theta) \tag{A.12}$$

$$y = z + \tan(\phi) \tag{A.13}$$

Substituting Equation *A.12* and Equation *A.13* into Equation *A.11* the following equation can be derived:

$$r^2 = z^2 \left(1 + \tan^2\theta + tan^2\phi\right) \tag{A.14}$$

With these equations we can now solve for x, y, and z in terms of the values r, θ, and ϕ returned from the range scanner. This gives the following:

$$x(r,\theta,\phi) = \frac{r \tan\theta}{\sqrt{1 + \tan^2\theta + tan^2\phi}} \tag{A.15}$$

$$y(r,\theta,\phi) = \frac{r \tan\phi}{\sqrt{1 + \tan^2\theta + tan^2\phi}} \tag{A.16}$$

$$z(r,\theta,\phi) = \frac{r}{\sqrt{1 + \tan^2\theta + tan^2\phi}} \tag{A.17}$$

With these equations, the range image, and the given field of view for the range scanner, the (x, y, z) coordinates of the scanned scene can be recovered. Using these concepts, a simulated scanner has been developed and is described in Section *A.7*.

For visualization of the range data we use a mesh of triangles generated from the range image. The points in the range image are arranged on a structured grid so connecting points will be adjacent to each other. From this grid we then create zero, one, or two triangles from four points in the range image that are in adjacent rows and columns [107]. This is done by first finding the shortest diagonal between the points. If this is below a preset threshold then two possible triangles may be formed from the two sets of three vertices. Each of these will then form a triangle if the edge lengths fall below a preset threshold.

A.7 Simulated Range Scanning

Using the concept of an orthogonal-axis angle range sensor as described in Appendix *A.6*, a simulated scanner has been developed to aid in the development of the modeling software (see Figure *A.11*). With the simulated scanner, range images of objects can be made quickly without having to set up all the hardware needed for a real scan. For testing, a simulated model may be all that is needed. This simulator allows the user to set the horizontal and vertical scan angle ranges along with the resolution of the synthetic image generated using sliders. The input is an inventor file either supplied on the command line or entered from a dialog from the *File* menu. A standard Open Inventor Examiner Viewer is used to set the view that the user wants to range using the mouse. The range image is then output using a dialog from the *File* menu in either standard 8-bit

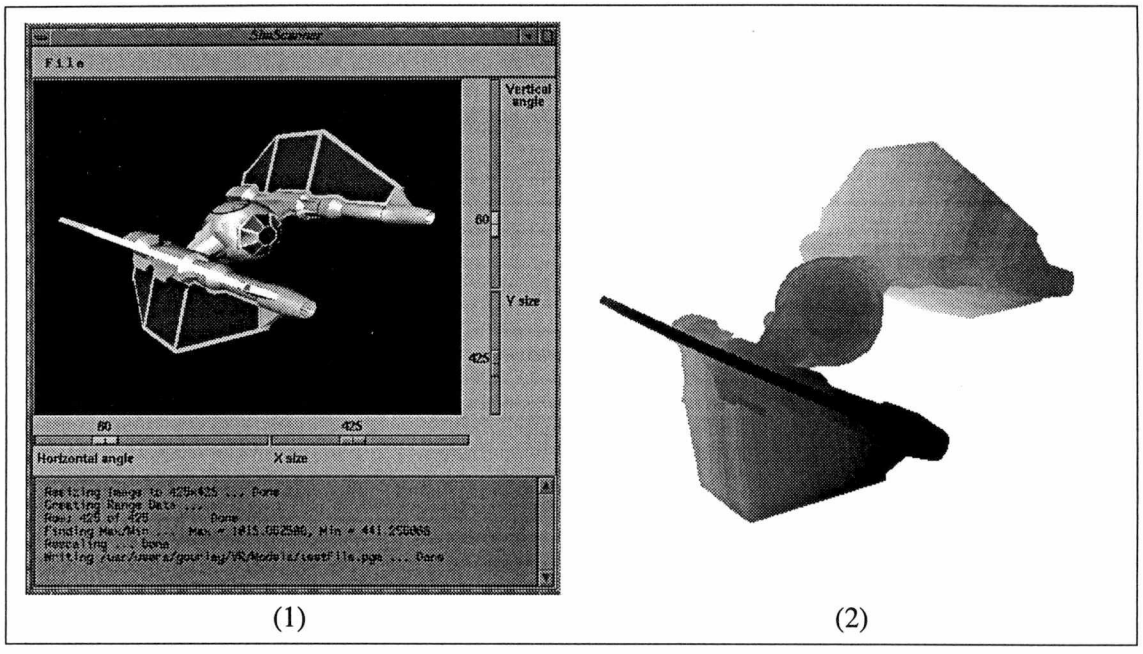


Figure A.11: *(1) Simulated orthogonal-axis scanner user interface and (2) an output range image from the simulated scanner.*

PGM format with the data scaled or or a FITS floating point format. The header of the PGM file contains information regarding the horizontal and vertical angles, maximum and minimum values used for scaling, and the camera transform of the camera view used to make the range image. Current status of the interface and feedback is given to the user in a dialog at the bottom of the interface. The result of a simulated scan is shown in Figure *A.11*.

APPENDIX B

Virtual Reality

This research falls into the broad area of visualization, which in recent years has begun to overlap with virtual reality. While the primary focus has been mesh reduction for the display of large data sets, the use of virtual reality for visualization has also been an issue. Therefore, in this appendix an overview of virtual reality is given.

Virtual Reality (VR) is a relatively new field and still maturing. Much of the technology does not live up to the media hype that has surrounded it. This chapter gives a background into what VR equipment is available and which areas of research this equipment is being used. Our application will be in visualization where the VR equipment is used as a method for 3D interaction and display.

B.1 Virtual Reality Overview

The precise definition of VR is difficult because of the many subjective applications and connotations associated with the VR field. To start with, the term *virtual reality* was first used by Jaron Lanier [62] in 1984 to describe "an immersive, interactive simulation of realistic or imaginary environments." Myron Krueger had coined the term *artificial reality* earlier in 1974 to describe computer controlled non-intrusive environments [58, 59]. But the idea of VR was first presented much earlier in 1965 by Ivan Sutherland.

> The ultimate display would, of course, be a room within which a computer can control the existence of matter. ... With appropriate programming such a display could literally be the Wonderland into which Alice walked. [100, 101]

The development of the technology had its beginnings within NASA and the military for use in flight simulators. Although these ideas existed and some limited research was being conducted, VR did not reach mainstream until June 1989 when the first head-mounted displays and datagloves

were demonstrated at trade shows. As computer technology develops, the ideas set forth years ago are now becoming reality.

Many different definitions of what constitutes VR are available. Variations also exist such as virtual environments (VE), synthetic environments (SE), virtual worlds, artificial reality (AR), augmented reality, and enhanced reality. Pratt [76] makes no distinction among a virtual worlds, virtual environments, or virtual reality. According to this definition a VE has three major elements:

1. Interaction – inputting and receiving data from the system,
2. 3D graphics – computer output that the user sees, and
3. Immersion – the feeling of presence.

Although these definitions can encompass a wide variety of applications individually, what creates the VE is combining all three together in an application in real-time. One good definition is:

> Virtual Reality is a way for humans to visualize, manipulate and interact with computers and extremely complex data. [5]

The term augmented reality does describe a slightly different context. Here virtual data and real data are shown to the user at the same time. Another VR definition is:

> ... a concept essentially deal[ing] with convincing the participant that he is actually in another place, by replacing the normal sensory input received by the participant with information produced by a computer. This is usually done through three-dimensional graphics and I/O devices which closely resemble the participant's normal interface to the physical world. The most common I/O devices are gloves, which transmit information about the participant's hand (position, orientation, and finger bend angles), and head-mounted displays, which give the user a stereoscopic view of the virtual world via two computer-controlled display screens, as well as providing something to mount a position/orientation sensor on. [45]

Jaron Lanier has recently defined virtual reality as:

> the use of technology to generate the sensory experiences of people, under human control. ... your experience of it is with the natural language of your own body. [10]

Summarizing all of the above definitions and explanations, we will define virtual reality as follows:

> *a means by which a user can intuitively interact with the computer via 3D graphics and icons in real-time to accomplish a task with a degree of realism.*

This definition blankets all of the previous definitions of virtual reality, artificial reality, augmented reality, and the like. It is still somewhat purposely vague. For example, the degree of realism does not need to be great, but it does need to be there. Therefore, an HMD does not have to be used to provide stereoscopic images, but a simple computer monitor can be used for display purposes or a large screen display.

The current technology cannot live up to all the hype given to VR by the media. "... it must be concluded that Virtual Reality still is a matter much for the research laboratories. The performance of current systems is still not good enough to enable useful work to be done [1]." Research in VR and using the tools and hardware developed for VR is just beginning but rapidly growing. VR represents the state-of-the-art in both computer hardware and software, pushing both to their limits and beyond. In fact "movement beyond the research lab has highlighted deficiencies – and spurred an urgent desire for improvement [2]." Current performance is marginal and is at the boundary of useability, thus the need for display tricks such as multiresolution models. The VR field has died down over the past few years. Many of the small companies that sprung up to build VR hardware have since gone out of business because the expectations from all the hype could not be met. VR, however, is a means by which the computer and human user can interact.

B.2 Virtual Reality Hardware

The hardware used for VR is a key ingredient in the human-machine interface (HMI). It consists of all the devices used to present information to the operator, and to pass information from the operator and sensors to the machine. Currently, there is a large gap between state-of-the-art commercially available hardware used for VR tasks and the hardware that is needed to create the

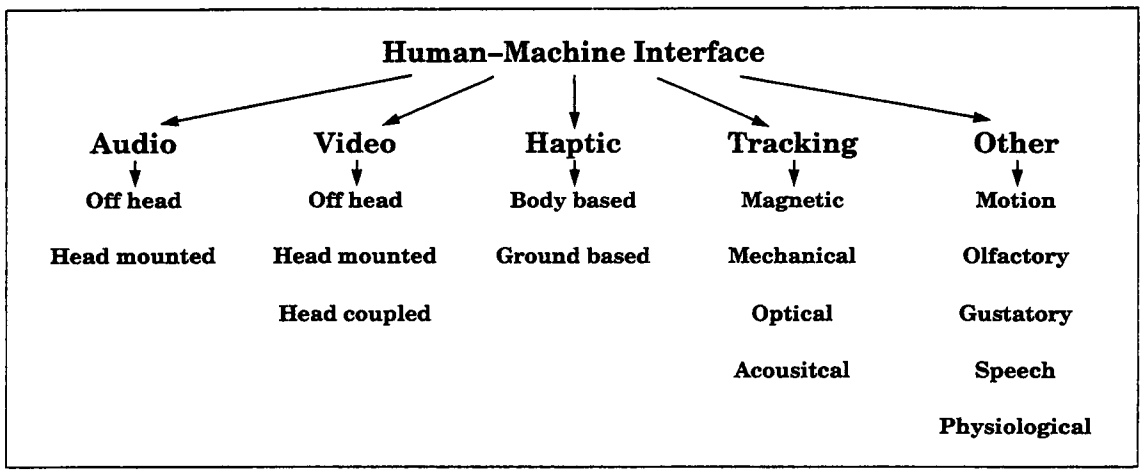


Figure B.1: *Breakdown of HMI hardware from VR technology.*

potential interface between human and machine [70]. Currently, VR hardware setups are mainly task driven [8]. Hardware used for VR can be loosely divided into that for interfacing of

- audio,
- video,
- haptic, and
- tracking.

Some devices have the capability of interfacing to more than one of these categories. A VR setup is known as a pod and will typical consist of a piece of hardware from each of these categories, such as an HMD, dataglove, Polhemus tracker, and speakers all connected to a computer. Each of these categories can be further subdivided. A breakdown of the hardware hierarchy is shown in Figure *B.1*. The remainder of this section gives a general overview of the hardware that is currently available for VR applications.

B.2.1 Video Interface

Visual displays are available in three forms: head-mounted displays (HMD), off-head displays (OHD), and head-coupled displays (HCD). Much recent development has been put into visual dis-

plays with the advent of high resolution video displays for computer graphics and HDTV. HMD's however, still suffer from low resolution and small fields-of-view (FOV) because of the low weight and small display size requirements.

Head Mounted Displays

Typical resolution of current popular HMD's is 320x240 pixels, typical TV resolution. For the FOV which is associated with human vision, this gives a resolution to the user that is legally blind [2]. Resolutions on the order of 2048x2048 are needed for a wide FOV and 20/20 vision. To compensate for this shortcoming, many HMD's limit the FOV. Figure *B.2* gives a comparison of some of the more popular HMD's in use today. Most of the lower-end HMD's rely on LCD displays, while the higher-end models use CRT's routing video to the eye via fiber optics giving much higher resolution.

Low-end HMD's have been created mainly for entertainment and can be purchased for as low as $100. Some of the lower end models only use one display for both eyes so a true stereo effect is not possible, but tracking is possible to get a spatial feel. One such device is the Stunt Master by VictorMaxx which sells for $50. Virtual I/O's i-glasses are unique in that they are the lightest HMD available, weighing only eight ounces [83]. This HMD limits the FOV to 30 degrees to give an increase in the perceived resolution. Semi-silvered mirrors also give the i-glasses a semi-HUD effect. Several popular HMD's include the CyberMaxx, the Forte Technologies VFX1, the Virtual Research HMD, and the Kaiser 1000pv. High-end HMD's have been driven mostly by the military. These use CRT's which route the video to the user's eyes via fiber optics. They have high resolution, wide FOV and can cost up to $1 million. Many HMD's also are also equipped with headphones and a tracking device. Some also have built in microphones. The major disadvantage of an HMD is that only one person can use it at a time. For multiple users to share the same VE, multiple HMD's must be used.

Off Head Displays

The OHD's are used in conjunction with more traditional CRT's and computer displays. Input to the operator from these can come in various forms. One simple method is by the use of red/green

Model	FOV (degrees)	Resolution (pixels)	Price	
Virtual-i/o i-glasses	30	263x230	$800	
CyberMaxx	56	267x225	$900	
Forte VFX1	48	278x204	$1,000	
Kaiser VIM 1000PV	100	710x225	$8,000	
Virtual Research VR5	42.4	640x480	$14,000	
Virtual Reality Inc. 133	116	2560x1024	$64,000	

Figure B.2: *HMD comparison.*

stereo glasses. In this case the separate video channels are displayed with different colors and visually combined by the operator.

Also, shutter glasses are available which allow the different images to be displayed at high rates and only one eye views each image. OHD's include the Crystal Eyes, 3D-Max, an entry level model costing $180, and the Sega glasses. Shutter glasses are typically much cheaper than HMD's since they rely upon an external monitor to provide the video source. However, extended use of shutter glasses has been known to produce headaches.

A computer monitor itself can be used as an OHD. Although no stereo information can be effectively displayed and a strong sense of immersion is not felt, useful information can be given to the operator.

Head-coupled display

An example of a head-coupled display is the Binocular omni-orientational monitor (BOOM) from Fakespace Inc. (see Figure *B.3*). Here the high resolution CRT display is mounted and supported on a mechanical arm so the operator can move the display to get varying views. The user gets much higher resolution, but is more limited physically. The use of the BOOM also gives the operator quick entry and exit to the VE versus an HMD. Also, a desktop model known as the PUSH is available.

B.2.2 Audio Interface

Audio interfacing can also be divided into two categories: head-mounted audio (HMA) and off-head audio (OHA). The communication of audio information occurs via speakers and microphones. Speakers can be in the form of headphones or loudspeakers. The current hardware available for audio is satisfactory for most VR applications. Inadequacies occur mainly in the spatial localization of the sound. Most sound systems, including headphones, only have two speakers which cannot give the perception of a sound located at a specific point. Arrays of up to thirty speakers surrounding the user have been tested to provide spatial location of sound. Emerging entertainment technology such as surround sound can help in producing correctly place sounds. The Musical Instrument Digital Interface (MIDI) is a standard by which music and sounds can be

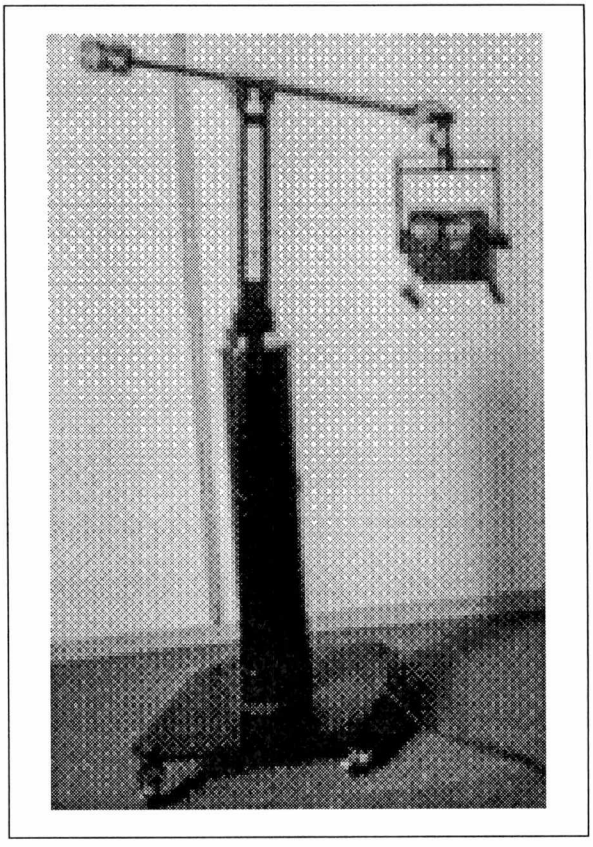


Figure B.3: *BOOM.*

controlled. Either a MIDI controller or a computer can be used to accomplish this task. When used with a computer in a VR environment, MIDI is a way to control the production of sound. Echo, pan, volume can all be controlled in real-time to help give a sense of spatial localization.

B.2.3 Haptic Interface

A haptic device is one that provides touch, tactile, or force feedback to a user. Haptic interfaces can be divided into two main categories: body-based devices and ground-based devices.

Body-based

The *Cyber*Glove from Virtual Technologies is a high-end data glove. The system includes a glove equipped with 22 bend sensors, each with 8-bit resolution, along with a control interface for serial communication. More about the *Cyber*Glove is given in Section *2.3*.

A popular cheap data glove is the Mattel PowerGlove [103] used originally for video games. This glove was scheduled for re-released with a serial interface in 1996 for $120. This glove also uses variable resistance strips (one per finger) and gives two bit resolution.

VPL's data glove [63] utilizes flexion technology in which specially treated fiber optics are used to detect bends. As the fiber is bent, the light passing through the fiber is attenuated. This glove has ten flex sensors (two per finger). The 5^{th} Dimension glove [83] also uses fiber optic bend sensors. This glove costs $500 and has one sensor per finger. The Dexterous Hand Master (DHM) is an exoskeleton that was originally used as the master for the dexterous hand from Utah/MIT. This unique glove uses Hall-effect sensors to detect bends. It has twenty sensors, four per finger.

Ground-based

A popular ground-based input device is the Spaceball costing about $1500. This device uses strain gauges to measure six DOF. More about the Spaceball is give in Section *2.3*. Another ground based input device is the Immersion probe. The Logitech Cyberman [82] is a low-end, 6-DOF input device with limited tactile feedback. The device has low resolution, providing seven bits of information in X and Y and only two bits for the remaining 4-DOF. However, this device only costs about $100. This device has been criticized for its poor ergonomics [24].

B.2.4 Position and Tracking Interface

There are four basic categories that position trackers fall into: magnetic, optical, acoustical, and mechanical [7]. Magnetic trackers can be either AC or DC based. The AC based trackers use three mutually perpendicular electromagnetic coils. An AC signal is applied to these coils and the resulting induced currents in the receiver measured. The Polhemus line of trackers is AC based. DC based trackers work on the same principle except a DC pulse is applied to the coils. The Ascension Flock of Birds is DC based. The Polhemus Fastrak® is an electromagnetic, six-degree-of-freedom tracking instrument. This includes a control unit that connects to a transmitter and up to four receivers and also has a port for serial communication. The tracking system employed by the Polhemus Fastrak uses electromagnetic fields to determine the position and orientation of an object. Polhemus recently announced that the Polhemus Insidetrak will be the first true 6-DOF

tracker for under $1000. More about the Fastrak is given in Section 2.3.

There are two types of acoustic trackers: time-of-flight (TOF) and phase coherent. The TOF trackers use three ultrasonic transmitters and receivers. By timing the signals from the various transmitters to the receivers the location is found. Phase coherent trackers compare the phase of the transmitted signal to the received signal to determine position. Optical based trackers are usually beacon based. In this case, the beacons are tracked by a fixed camera or fixed beacons are tracked by a moving camera. Also, laser ranging systems can be used for tracking. Mechanical trackers have a movable arm, like the master controller in a teleoperation system.

B.2.5 Other Interface Hardware

The Bodysuit is similar to a dataglove, but it is for the entire body. Currently only one bodysuit has been built. It is fiber-optic sensor based. Other sensors include those for olfactory (smell), gustatory (taste), speech, and physiological inputs. Speech synthesis and recognition are very important in everyday communication between humans. These are linked with the audio interface since the microphones and speakers are the means of interaction. Physiological interfaces are associated with such things as brain activity, muscles stimulation, and respiratory system. Practical use of these signals is still several years away.

B.3 Applications of Virtual Reality

Now that we have discussed the hardware that is available for performing research in VR, what are the application areas to which this technology can be applied? These areas are highly varied. A current hotbed for low-end VR is home entertainment. VR has also found its way into research areas as a new tool, which is where we will be using it. VR can be defined to include a wide range of applications. Some definitions include such things as teleconferencing, multimedia computing for education, and internet surfing. In this section we focus on the current research and applications that are on-going utilizing VR technologies in a more scientific manner. These may include the use of single pieces of VR equipment to increase productivity and performance, or they may be complete immersive environments. Application areas include:

1. Entertainment and artistic applications [4]
2. Data or model visualization [93, 74]
3. Designing, planning, and manufacturing [99, 40]
4. Education and/or training [105, 13]
5. Teleoperation and hazardous operations [14, 98]
6. Psychological test beds [47]
7. Communication and collaboration
8. National defense [68]
9. Medicine and health [72, 112, 36]

We are applying VR to data visualization of range data sets. In visualization applications "the image presented to the user is a combination of a computer-generated image and the view of the real world around the user." [81] For psychological and perceptual requirements, interactive response times of 0.1 seconds or less are needed along with at least 10 frames/second update rates. Also high-quality, high-resolution (at least 1000 x 1000), full-color displays with a wide field-of-view are needed. HMD's have not been well excepted by researchers for these reasons. Much of the research done in data visualization has been done in the field of medicine. Medical visualization requires that very large amounts of graphic data be displayed. These data sets can be acquired in a variety of ways, including MRI, PET, CT, and ultrasonic scanners. Design and manufacturing applications utilizing VR are still in the development stages. Distributed and custom manufacturing will greatly benefit from VR. Education and/or training using VE creates a cost-effective way using simulations instead of training people in a real world situation. Teleoperation of robots was developed mainly for hazardous operations, such as in nuclear plant or underwater application. Psychological test beds utilizing VR during treatment of patients are appearing. National defense is pushing high-end VR. The flight simulators used to train pilots is a good example of

what has been done in this area. Medicine and health currently has a huge push in VR. Teleoperation and virtual patience are receiving much attention. Much of this work, however, is still in the development stages.

The key components of any visualization system are independent of the application. This is the VR engine. Most visualization which is going to involve VR usually does so for one main reason, the data set is three dimensional in nature, otherwise traditional methods would suffice. For this reason, application developed to visualize one set of 3D data should be useful for other types of 3D data. In particular, the techniques for placing multiple sets of 3D laser range data together into one large data set could be used for multiple sets of medical data obtained from differing views using a PET or MRI scanner.

B.4 Glossary of Acronyms

ANN	Artificial neural network
AR	Artificial reality
BOOM	Binocular omni-orientational monitor
CAVE	Cave Automatic Virtual Environment
CHI	Computer-human interface
CRT	Cathode ray tube
DHM	Dexterous hand master
DIP	Distal interphalangeal
DOF	Degrees-of-freedom
FFB	Force feedback
FOV	Field of view
GUI	Graphical user interface
HCD	Head-coupled display
HDTV	High definition television
HMA	Head-mounted audio
HMD	Head-mounted display
HUD	Heads-up display
HMI	Human-machine interface
IP	Interphalangeal
IVE	Interactive virtual environment
LCD	Liquid crystal display
MCP	Metacarpophalangeal
MIDI	Musical Instrument Digital Interface
OHA	Off-head audio
OHD	Off-head display
PE	Processing element
PIP	Proximal interphalangeal
SE	Synthetic environment
SGI	Silicon Graphics, Inc.
SUI	Spatial user interface
TFB	Tactile feedback
TOF	Time-of-flight
VE	Virtual environment
VR	Virtual reality
VRD	Virtual retinal display
VW	Virtual world

B.5 Glossary of Terms

Artificial reality: Introduced by Myron Krueger in the mid-1970's describing non-intrusive environments controlled by computers. (see Virtual reality)

Augmented reality: Use of HUD to overlay computer generated data onto the real environment.

Autonomous: Control scheme in which a robot depends upon preprogrammed algorithms for all decisions.

Binocular omni-orientational monitor (BOOM): An option to HMD's. This HCD is a 3D display device that is suspended from a mechanical boom which swivels.

Cave Automatic Virtual Environment (CAVE): VE using projections onto the walls and ceiling to give the feeling of immersion.

Computer-human interface (CHI): See Human-machine interface.

Cybernetics: Study of the interaction between humans and machines

Cyberspace: First use in *Neuromancer* by William Gibson in 1984 to describe a shared matrix in the world's computer network.

Data glove: A glove wired with sensors to recognize hand gestures for use in interaction with VE's.

Data suit: Similar to a data glove, but for the entire body. Only one has been built.

Degrees-of-freedom (DOF): Number of independent translations and rotations.

Enhanced reality: See Augmented reality.

Field-of-view (FOV): Angle of the view port for a scene

Force feedback (FFB): Simulation of weights and resistances to produce a force that can be felt.

Gesture: Physical positions or movements used to convey information.

Haptic: Touch (tactile) and force feedback.

Head-coupled display (HCD): Stereo display device which is mounted to a mechanical tracker

Head-mounted display (HMD): Stereo display device that has tiny monitors mounted in front of each eye.

Heads-up display (HUD): A see through HMD that overlays computer generated data onto the real environment.

Human-machine interface (HMI): Hardware and software that interfaces between the operator and the computer.

Immersion: The feeling of presence in the VE; the degree of realism.

Kinesthetic: Position and movement perceived through joints.

Musical Instrument Digital Interface (MIDI): Standard to interface musical instruments which can be used for spatial audio.

Pod: Virtual reality hardware setup.

Posture: A static gesture.

Presence: The feeling of "being there."

Shutter glasses: Liquid crystal glasses which present different images to each eye by covering each eye alternately.

Spaceball: A stationary force/torque sensor used as a 6-DOF input device.

Synthetic environment (SE): Virtual environment for simulation.

Tactile feedback (TFB): Sensations applied to the skin, such as Braille.

Teleautonomous: Control scheme in which a robot uses both preprogrammed algorithms and human input for decisions.

Telepresence: The ability to view, control, and interact with physically distance environments.

Telerobotics: Control scheme in which a robot depends upon human input for all decisions.

Virtual: Not real, existing in essence or effect.

Virtual environment (VE): Realistic simulations of interactive surroundings. (see Virtual reality)

Virtual reality (VR): An immersive, interactive simulation of realistic or imaginary environments.

Virtual retinal display (VRD): A display which scans images directly onto the retina via lasers.

Virtual world (VW): Simulated models used to immerse a user

Visualization: Use of computer graphics to present data to the user.

VITA

Christopher S. Gourley was born in Sweetwater, Tennessee, on August 10, 1970. He received the Bachelor of Science degree in electrical engineering from the University of Tennessee, Knoxville in May 1992 with an emphasis in digital design and image processing. In May 1994 he received the Master of Science degree in electrical engineering from the same university with a research emphasis in robotics and tele-autonomous systems. He will be presented a Doctor of Philosophy degree in electrical engineering in December 1998. The research for this degree focuses upon the visualization of photo-realistic 3D models. His current research interests are in the fields of computer vision, robotics, and virtual reality. He is a member of Eta Kappa, Tau Beta Pi, Phi Kappa Phi, and IEEE.