

THESIS

OBTAINING 3D SILHOUETTES AND SAMPLED SURFACES FROM SOLID  
MODELS FOR USE IN COMPUTER VISION

Submitted by

Mark Richard Stevens

Department of Computer Science

In partial fulfillment of the requirements

for the degree of Master of Science

Colorado State University

Fort Collins, Colorado

Fall 1995

COLORADO STATE UNIVERSITY

August 31, 1995

We hereby recommend that the thesis OBTAINING 3D SILHOUETTES AND SAMPLED SURFACES FROM SOLID MODELS FOR USE IN COMPUTER VISION prepared under our supervision by Mark Richard Stevens be accepted as fulfilling in part requirements for the degree of Master of Science.

Committee on Graduate Work

---

Committee Member

---

Committee Member

---

Adviser

---

Department Head

## ABSTRACT OF THESIS

### OBTAINING 3D SILHOUETTES AND SAMPLED SURFACES FROM SOLID MODELS FOR USE IN COMPUTER VISION

Model-based object recognition algorithms identify modeled objects in a scene by relating stored geometric models to features extracted from sensor data. This process can be combinatorially explosive as the amount of information presented to the recognition algorithm increases. This thesis presents a method for extracting only relevant features from a stored three dimensional (3D) model in an attempt to reduce the difficulty of the recognition process. The development of the methods presented here were driven by the needs of the Automatic Target Recognition (ATR) algorithm being developed concurrently at Colorado State University (CSU).

The ATR algorithm locates an object using multi-sensor data by determining the correspondence between a range (LADAR) image, a color image, a thermal (FLIR) image, and a Computer Aided Design (CAD) geometric model. The success of this process depends in part on which features can be automatically extracted from the model database. Since the models available for this process contain more detail than is needed by the ATR algorithm, they must first be reduced to a more appropriate form.

From the reduced model, we can extract 3D point-sampled surface information as well as 3D model silhouette features. These features are used by the ATR algorithm to refine a pose estimate of the model relative to the sensors. The pose then provides a basis for measuring the quality of the match between 3D model features and sensor features.

Mark Richard Stevens  
Department of Computer Science  
Colorado State University  
Fort Collins, Colorado 80523  
Fall 1995

## ACKNOWLEDGEMENTS

As with any major accomplishment in one's life, it is customary to pay tribute to those who aided in its completion. I would like to begin by thanking my parents, both Elizabeth Tyrol, and Ronald Stevens who both taught me the value of a good education. I would like to thank my grandfather, Richard G. Colby, for his support and pride in the career choices I have made. I would also like to thank all of my siblings: Janet, Mark and Patty. I would also like to thank Soraya Rana for her emotional support and reassurance that I have made the correct career path.

Academically I would like to thank J. Ross Beveridge, for providing the guidance and infinite wisdom that all true mentors possess. Michael Goss for teaching me what I needed to know to complete this thesis. I would also like to thank Drew Schwickerath for his work on the ATR algorithm, and its eventual integration with the work presented in this thesis.

This work was sponsored by the Advanced Research Projects Agency (ARPA) under grant DAAH04-93-G-422, monitored by the U. S. Army Research Office.

## CONTENTS

<b>1 Introduction</b>	<b>1</b>
1.1 Overview	1
1.2 System Overview	4
1.3 Thesis Overview	4
<b>2 The Automatic Target Recognition Algorithm</b>	<b>6</b>
2.1 Overview	6
2.2 Phase I: Color Detection	7
2.3 Phase II: Hypothesis Generation	8
2.4 Phase III: Co-Registration	9
2.4.1 3D Silhouette	12
2.4.2 3D Sampled Surface	13
<b>3 RangeView: Visualization and Verification of Multi-sensor Object Recognition</b>	<b>17</b>
3.1 Overview	17
3.2 RangeView: A Tool for Multi-Sensor Visualization	17
3.3 RangeView Environment	18
<b>4 Sensor Data</b>	<b>21</b>
4.1 Overview	21
4.2 Sensor Comparison	21
4.3 Image Groups	22
4.4 Three Dimensional Range Images - LADAR	22
4.4.1 Sensor Characteristics	22
4.4.2 Calibration	26
4.5 Color Images - 35 mm	28
4.5.1 Sensor Characteristics	28
4.5.2 Calibration	29
4.6 Forward Looking Infrared Images - FLIR	29
4.6.1 Sensor Characteristics	29
4.6.2 Calibration	29
<b>5 Converting CSG to Polyhedra</b>	<b>30</b>
5.1 Overview	30
5.2 Explicit Versus Implicit Model Representations	31
5.3 Model Reduction	32
5.4 Model Conversion	34
5.4.1 Related Research	34

5.4.2	BRL/CAD Models . . . . .	37
5.4.3	The Conversion Algorithm . . . . .	40
5.4.3.1	Phase I: Subdividing Objects . . . . .	41
5.4.3.2	A Splitting Example . . . . .	44
5.4.3.3	Phase II: Labelling Polygons . . . . .	45
5.4.3.4	A Labelling Example . . . . .	46
5.4.3.5	Phase III: Determining Which Polygons to Keep . . . . .	47
5.4.3.6	A Classification Example . . . . .	48
5.4.3.7	Merging Split Faces . . . . .	48
5.4.3.8	A Hybrid Approach . . . . .	49
5.4.3.9	Results on a Vehicle Model . . . . .	49
<b>6</b>	<b>Extracting 3D Model Features</b>	<b>51</b>
6.1	Overview . . . . .	51
6.2	Stored versus Demand Driven . . . . .	52
6.3	Related Research . . . . .	53
6.4	The Model Extraction Algorithm . . . . .	56
6.4.1	Determining Visible Faces . . . . .	57
6.4.2	Obtaining the Silhouette . . . . .	60
6.4.3	3D Sampled Surface . . . . .	62
6.4.3.1	LADAR Geometry . . . . .	64
6.4.3.2	Generating Sampled Surfaces . . . . .	64
6.4.3.3	Results . . . . .	66
<b>7</b>	<b>Future Work</b>	<b>68</b>
7.1	Overview . . . . .	68
7.2	Model Reduction . . . . .	68
7.3	Model Analysis . . . . .	69
7.4	Sensor Data Feature Extraction . . . . .	69
7.5	Integration with the ATR Algorithm . . . . .	69
<b>8</b>	<b>Conclusion</b>	<b>70</b>
<b>9</b>	<b>REFERENCES</b>	<b>71</b>
<b>A</b>	<b>Acronyms</b>	<b>75</b>
<b>B</b>	<b>Color Plates</b>	<b>76</b>

## LIST OF FIGURES

1.1	Sensors and Corresponding Model of a M113 (See Color Plate 1) . . . . .	3
1.2	Overview of Entire Process . . . . .	5
2.1	The Three Stage ATR Algorithm . . . . .	7
2.2	Color Detection - M113 APC (nov21108c) (See Color Plate 2) . . . . .	8
2.3	LADAR Probe of M113 (See Color Plate 3) . . . . .	8
2.4	A Co-Registration Example (See Color Plate 4) . . . . .	10
2.5	Sensor Relationships . . . . .	10
2.6	Fixed Translation of Sensors, Slight Rotations . . . . .	11
2.7	Model Silhouette Shown with Color Image (See Color Plate 5) . . . . .	13
2.8	Silhouette from Various Angles (See Color Plate 7) . . . . .	14
2.9	LADAR Imagery (See Color Plate 6) . . . . .	15
2.10	Sampled Surface from Various Angles (See Color Plate 8) . . . . .	16
3.1	The RangeView Screen (See Color Plate 9) . . . . .	19
3.2	Key to understanding the RangeView Screen . . . . .	19
3.3	Optical Imagery Texture Mapped onto 3D Range Data (See Color Plate 10) . . . . .	20
4.1	Imagery Group Number 1 of M113 APC (See Color Plate 11) . . . . .	23
4.2	Imagery Group Number 2 of M113 APC (See Color Plate 12) . . . . .	24
4.3	Imagery Group Number 3 of M113 APC (See Color Plate 13) . . . . .	25
4.4	Calibration Geometry for LADAR Device . . . . .	28
5.1	M113 APC Model . . . . .	31
5.2	Intersection of Two Cubes . . . . .	32
5.3	The B-rep for the Intersection of two cubes . . . . .	33
5.4	Regularized Set Operations . . . . .	35
5.5	BRL/CAD Primitives Supported . . . . .	38
5.6	Converted Primitives . . . . .	39
5.7	Segments formed by a line . . . . .	43
5.8	Splitting Alphabet . . . . .	44
5.9	A Splitting Example . . . . .	44
5.10	A Labelling Example . . . . .	47
5.11	A Classification Example . . . . .	48
5.12	A Merge Example . . . . .	49
5.13	The M113 APC B-rep (See Color Plate 13) . . . . .	50
6.1	Assign a Unique Color to Each Face (See Color Plate 14) . . . . .	58
6.2	Color Table for Cube example (See Color Plate 15) . . . . .	59
6.3	Rendering of Cube (See Color Plate 16) . . . . .	60
6.4	Silhouette of the Cube (See Color Plate 17) . . . . .	61

6.5	Silhouette for M113 APC (See Color Plate 18) . . . . .	63
6.6	LADAR Geometry . . . . .	65
6.7	Sampled Surface for M113 APC (See Color Plate 19) . . . . .	67



## LIST OF TABLES

5.1	Splitting Two Objects . . . . .	42
5.2	Labelling a face . . . . .	46
5.3	Faces to retain for ObjectA . . . . .	47
5.4	Faces to retain for ObjectB . . . . .	47

## Chapter 1

### INTRODUCTION

#### 1.1 Overview

Model-based object recognition algorithms identify modeled objects in a scene by relating stored geometric models to features extracted from sensor data [Pop94]. A common goal of these techniques is to determine the position and orientation of the object relative to the sensor. This task is accomplished by aligning a three dimensional (3D) model of the object relative to the observed sensor data. In this way, the accuracy of the match between the object model and the sensor data can be compared. Accomplishing this task requires that relevant and comparable information be extracted from both the sensor data as well as the model database.

Proper choice of model representation is essential if features appropriate for matching are to be efficiently extracted from the model database. While the field of solid modeling has made numerous advances in both the representation of geometry and topologies of solids in the last few decades, most geometric modeling schemes are still designed for purposes other than automated recognition. These representations typically lack the information most needed by the automated recognition process [Bes88]. The work presented here focuses not only on the appropriate model representation to use for a specific model based vision domain, but also how to extract relevant information from that representation.

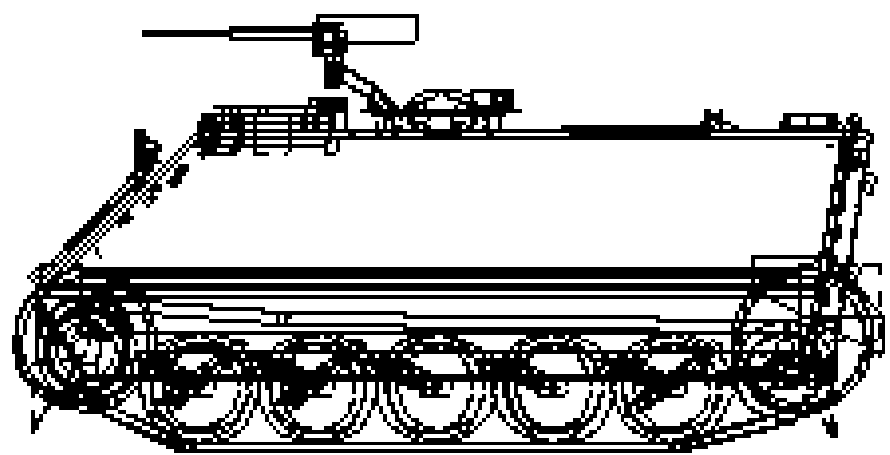
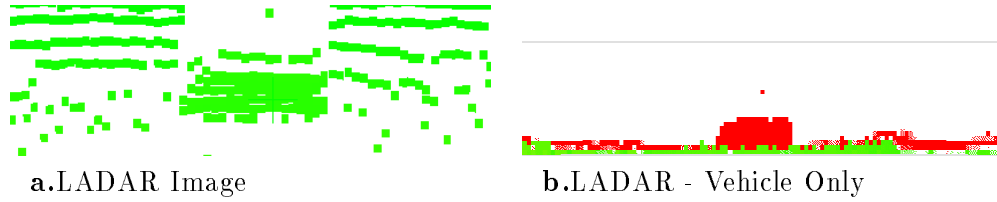
The models used by our automatic target recognition (ATR) process originate in the Ballistic Research Laboratory / Computer Aided Design (BRL/CAD) model format. The BRL/CAD format is an excellent example of a mature modeling scheme in which extremely detailed and complete information about a vehicle can be represented. However, due to the implicit nature of its Constructive Solid Geometry (CSG) format, explicit 3D vertex, edge,

and face information is not directly available. Therefore, extracting feature information from a CSG model can be computationally expensive. What we need is a model format which allows easy extraction of 3D face and edge information and as well as efficient removal of irrelevant features. For our application we have chosen to use a boundary representation which allows for easy retrieval of the information we deem comparable to our sensor data.

The sensor data used by the ATR algorithm consists of two types of optical data (Color and Thermal (FLIR)), and range imagery (LADAR). Figure 1.1 shows enlarged portions of data from the three sensor types juxtaposed with a high resolution BRL/CAD model. This simple example shows how little of the information present in the BRL/CAD model is evident in the sensor data. The model resolution is much higher than it needs to be for matching to data of this form. Moreover, this needless detail will serve to make the recognition process a more difficult task, not an easier one.

Model matching is a combinatorially explosive problem [Bev93]. The naive approach to model matching would be to determine a match error between all possible combinations of model and data features, and then choose the smallest error set as the optimal object match. However in a system where only 10 model features are being compared against 10 data features, there exist  $2^{100}$  possible combinations. Being able to reduce the number of data features to only 5 would reduce cut the correspondence space in half ( $2^{50}$ ). Therefore picking the appropriate model features for matching can greatly simplify the problem.

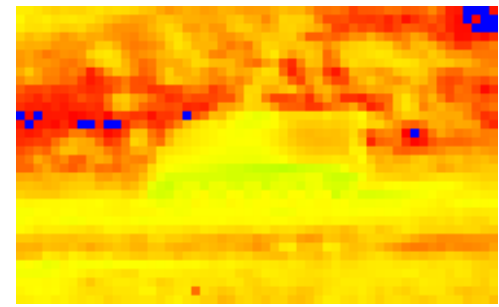
A better approach is to extract only relevant features from the model for a hypothesized viewpoint and compare only those features to the sensor information. Varying the viewpoint slightly will change the set of visible model features, and a new match error can be calculated. As the new error is determined, its relation to previously determined errors can be observed. A matching system can use this information to predict another viewpoint which may have an even lower match error. By guiding the changes in viewpoint in the direction of the minimal error, a matching system can converge to a solution in an acceptable amount of time. The utility of this approach is highly dependent on the features extracted from the model database. The work presented here focuses on how to



c.A BRL/CAD Model



d.Enlarged Color Image



e.Enlarged FLIR Image

Figure 1.1: Sensors and Corresponding Model of a M113 (See Color Plate 1)

extract model features which are directly comparable to the information present in the sensor images.

## 1.2 System Overview

A multi-tiered system to handle the various phases of the model conversion process, and its use by the ATR algorithm, has been developed. The models begin in the BRL/CAD format. Within the native BRL/CAD environment, a user can reduce the level of model complexity. The results of this process are then fed into the conversion algorithm. Automating this process is fairly complicated since it requires examining how each piece of the model relates to the overall model shape and appearance. In order to expedite our production of reduced models for use in ATR, most of the reduction effort lies in the hands of the user. However, some automated reduction occurs within the next phase of the system which converts these user-reduced models from CSG to a polyhedral representation.

The new polyhedral model is then stored in a database which is accessed directly by two different applications. The first is the model feature extraction module which generates relevant model features for a requested viewpoint, and provides that information to the ATR algorithm. The second application is called RangeView, which is a visualization and verification tool which allows the quality of the ATR results to be manually assessed. A graphical depiction of the process overview is shown in Figure 1.2.

## 1.3 Thesis Overview

This thesis is broken down into eight distinct chapters. The main contribution of this work is the model conversion and extraction phases presented in Chapters 5 and 6. However, before delving into these methods it is essential to understand the ATR process. Because the ATR algorithm is the driving force behind this work, Chapter 2 is presented in order provide a motivation for which features need to be extracted from the model and why.

Chapter 3 presents a tool, known as RangeView, designed to visualize the ATR results. This tool was the precursor to the methods developed in this thesis. Several of the images

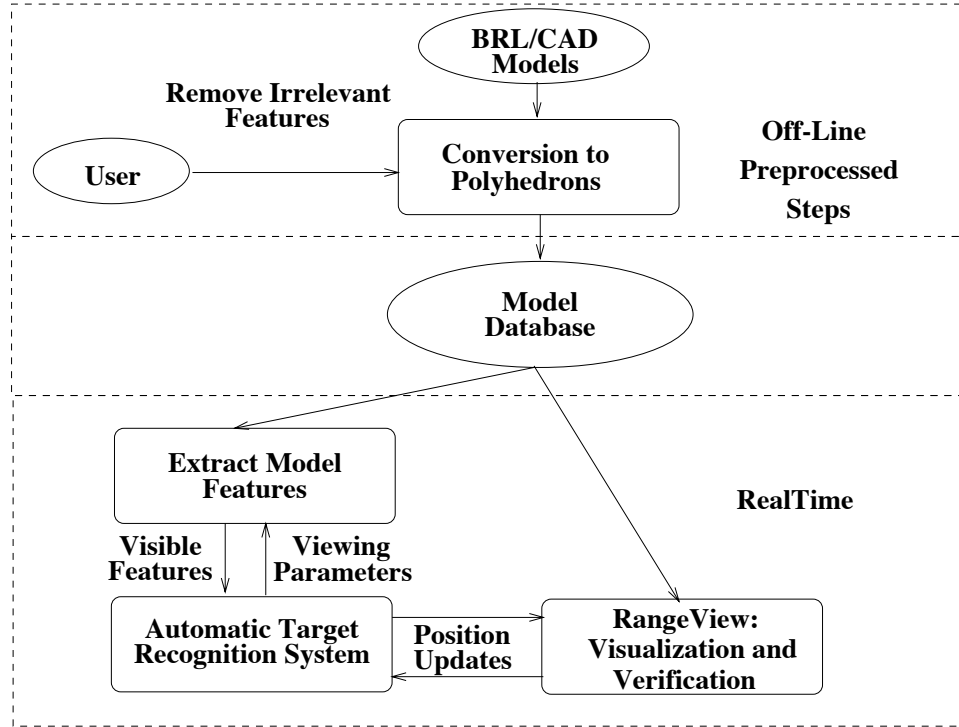


Figure 1.2: Overview of Entire Process

in this thesis were also generated using this tool, and the discussion centers around how to visualize all of the different sensors simultaneously. Chapter 4 takes a closer look at the sensor data and provides a detailed description of the level of detail present in our imagery. This chapter is essential for understanding how the model features relate to the sensor data, and the level of detail required in the model database. Chapters 5 and 6 present the actual model conversion and use. The final two chapters present future directions of this research and conclusions.

## Chapter 2

### THE AUTOMATIC TARGET RECOGNITION ALGORITHM

#### 2.1 Overview

The ATR algorithm is the driving force behind the development of the algorithms for extracting 3D features from the CAD models. While the ATR work is not the direct focus of this thesis, understanding this broader project is essential to understanding the context for the model feature extraction work which is the main contribution of this thesis. This chapter will review the general the goals of ATR algorithms, as well as the fundamental problems associated with working in this domain. Justification for many of the assumptions used in developing the model extraction routines are also discussed here.

The ATR system being developed at Colorado State University will attempt to locate military vehicles using input data from several different sensors. The project is a part of the Reconnaissance, Surveillance, and Target Acquisition (RSTA) software for the Advanced Research Projects Agency (ARPA) Unmanned Ground Vehicle (UGV). The data set used in the development of the algorithms was obtained in November of 1993 at Fort Carson, Colorado [BPY94]. Data from three sensor types was collected: color imagery, thermal (FLIR) imagery, and range (LADAR) information. Our main goal is to first determine if a vehicle is present in this imagery and second, if a vehicle is present, determine its optimal pose. An detailed overview of the entire process can be found in [BHP95, BHP94], and is reviewed below.

To accomplish this task, the ATR algorithm uses a three stage strategy [BHP94]. First, a detection process suggests regions of interest within the image worth further consideration as possible targets. An innovation at this stage, being developed at the University of Massachusetts, is the use of color as an additional detection cue [BDHR94].

The second stage, being developed by Alliant Tech Systems, extends LADAR probing techniques [BJLP92] to generate target type and target pose hypotheses. Finally, given object type and pose hypotheses, an error reduction approach will generate a best-fit match between the sensor and model features [SB94]. The final phase, being developed at CSU, is referred to as Co-Registration and is discussed in the Section 2.4. Figure 2.1 shows the three phases of the ATR work, as well as which institution is leading the development effort.

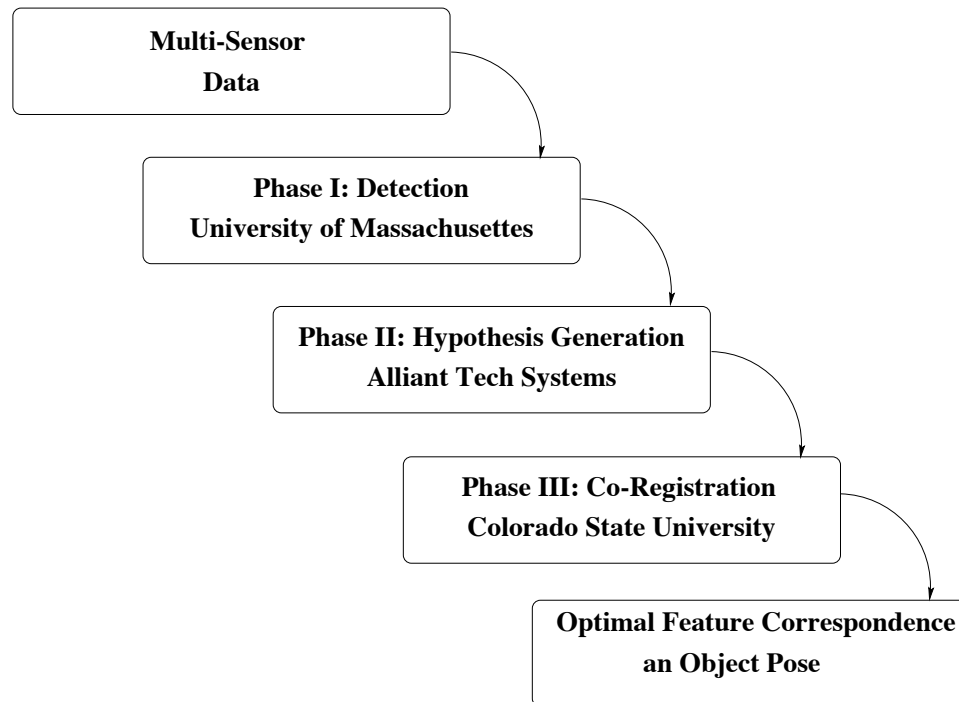


Figure 2.1: The Three Stage ATR Algorithm

## 2.2 Phase I: Color Detection

Phase I of our target detection system uses a learning algorithm to determine which colors in imagery are representative of military vehicles. The multi-variate decision tree algorithm is trained on a set of images which have had each pixel labelled as representing either vehicle or not vehicle. The image training set consists of vehicles of similar camouflage, and model as those to be detected. The images are also for the same type of terrain with similar types of vegetation. The learning algorithm identifies which colors are consistent with military vehicles, and uses this information to find regions of interest



in an image. Using photographic quality color imagery collected at Fort Carson [BPY94], it has detected 109 out 112 targets with 44 false positives. Figure 2.2 shows a color image along with the corresponding region of interest found by the color detection algorithm. This system can also generalize its findings to different types of weather and times of day.

In parallel, a thermal detection algorithm developed by Lockheed-Martin, looks for vehicles in the FLIR data. By studying thermal signatures of the military vehicles, it is possible to predict how they will appear in the FLIR image. This FLIR detection algorithm, along with the Color detection algorithm are performing fused Color and FLIR target detection on the Lockheed-Martin Unmanned Ground Vehicle. Combining these two techniques reduces the number of false positives decreases, and provides a more accurate initial detection.

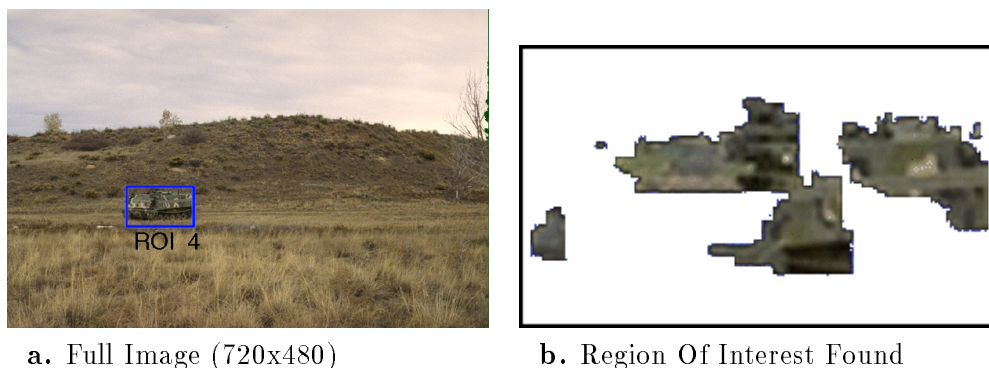


Figure 2.2: Color Detection - M113 APC (nov21108c) (See Color Plate 2)

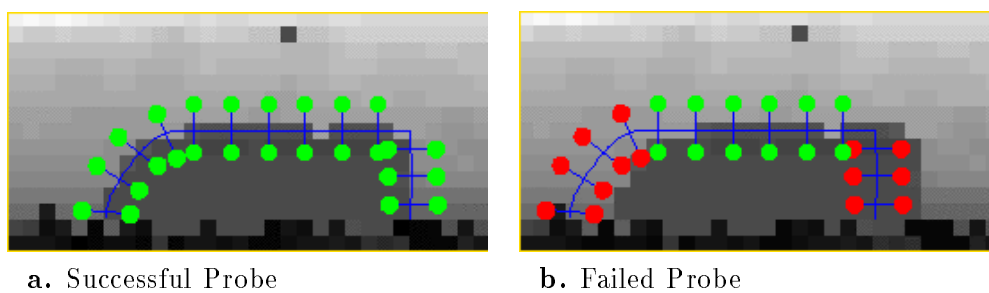


Figure 2.3: LADAR Probe of M113 (See Color Plate 3)

### 2.3 Phase II: Hypothesis Generation

Phase II takes the initial detection provided by Phase I, and examines corresponding positions in the LADAR data to determine several possible pose estimates for different

vehicle types. This process, known as hypothesis generation, provides Co-Registration with a set of possible vehicle models to use for matching in the imagery, as well as a set of vehicle pose estimates.

Many model-based vision systems assume the object being sought in the scene is known before the process begins [Bes88]. Phase II makes no assumptions other than that knowledge about the vehicle must be present somewhere in its database. Using a template probing technique, and the database of various vehicle templates, the most likely set of vehicles present in the image can be determined. Once the probing is complete, an initial pose estimate can be obtained [GJSL90].

The entire process uses a probing technique in which 2D pre-computed templates are matched against the data. Figure 2.3 shows a sample LADAR image with a vehicle template being applied. At uniform positions along the template, a probe is applied to either side. The green circles in Figure 2.3 represent successful probes to either side of the template, and red represents failure. Using a guided search of the database, based on the results of the current template probe, the best template for the scene can be determined. The template of the best match provides both the identification and initial pose hypothesis which is passed to Phase III of the ATR process.

## 2.4 Phase III: Co-Registration

Co-Registration has two distinct goals: the first is to determine the transformation needed to register the information in the various sensor images, and the second is to find a transformation to position the model in the scene. Figure 2.4 shows an example of the Co-Registration algorithm operating with a vehicle model, a LADAR image, and a color image. The model is shown with a poor initial estimate, and placed far above its correct position. The color imagery is shown texture mapped onto the range data <sup>1</sup>. Notice the depth values corresponding to the terrain are painted blue, or sky color. As the process begins, Co-Registration brings the model into the correct position and corrects the registration in sensor imagery.

---

<sup>1</sup>This technique is a component of the RangeView visualization tool discussed in Chapter 3

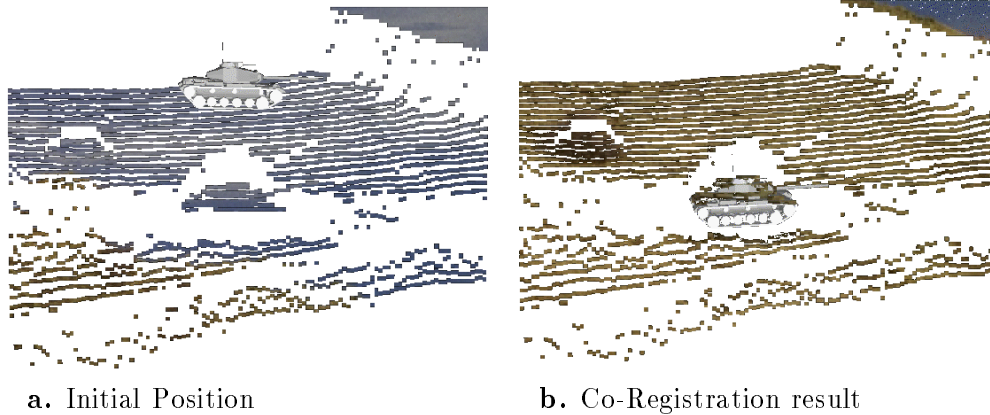


Figure 2.4: A Co-Registration Example (See Color Plate 4)

The three main sensors are nearly bore-sighted, but the alignment is not perfect and can vary slightly. Thus a slight mis-registration between sensor images may exist. Figure 2.5a shows the three different sensors nearly bore-sighted with a similar focal point in the direction of a target.

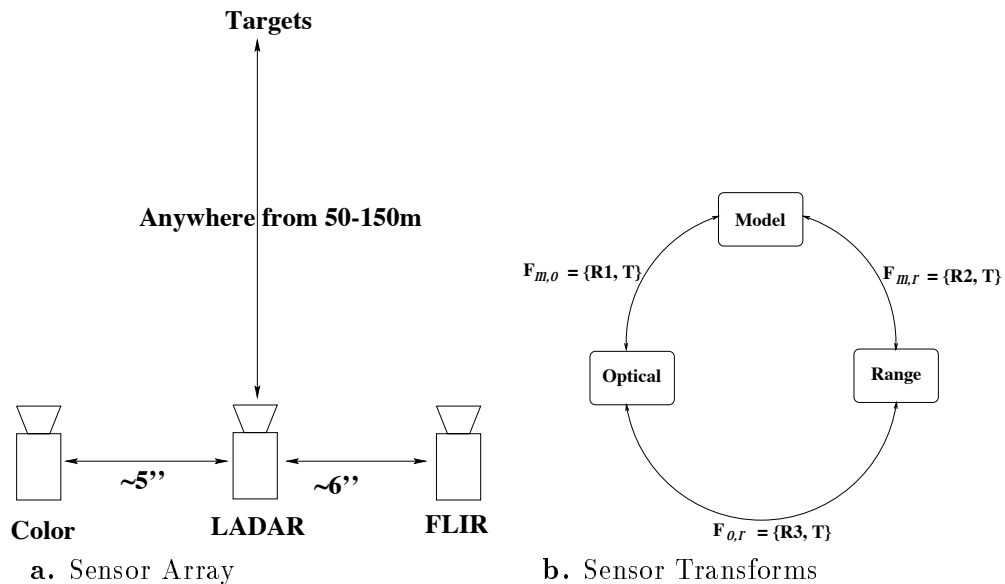


Figure 2.5: Sensor Relationships

Transformation matrices exist which allow placement of the model in the different sensor coordinate systems. Considering both the color and FLIR imagery as optical imagery, the existing set of transformations looks something like Figure 2.5b. The sensors in our data collection were firmly mounted a fixed distance apart, and therefore the transformation between them is known and considered fixed. However, the sensors are subject to

slight torsion rotations over time (See Figure 2.6). Therefore, slight inaccuracies exist in the rotation matrix which map between the optical and range imagery. The reconciliation, sometimes referred to as alignment, will register the different sensor images with respect to one another.

If two of the transformations in the loop (Figure 2.5b) are known, the the third link can be solved for automatically (i.e  $F_{m,r} = F_{m,o} F_{o,r}$ ). We have chosen to focus on determining the transformation between the model and the range imagery:  $F_{m,r}$ . Assuming  $F_{o,r}$  is perfectly known, this in turn allows the determination of the model to optical transformation:  $F_{m,o}$ . Therefore the transformations to and from the model coordinate systems ( $F_{m,r}$ ,  $F_{m,r}^{-1}$ ,  $F_{m,o}$ ,  $F_{m,o}^{-1}$ ) can constrain the relationships between the optical and range sensors so the  $F_{m,o}$  and  $F_{m,o}^{-1}$  can easily be determined.

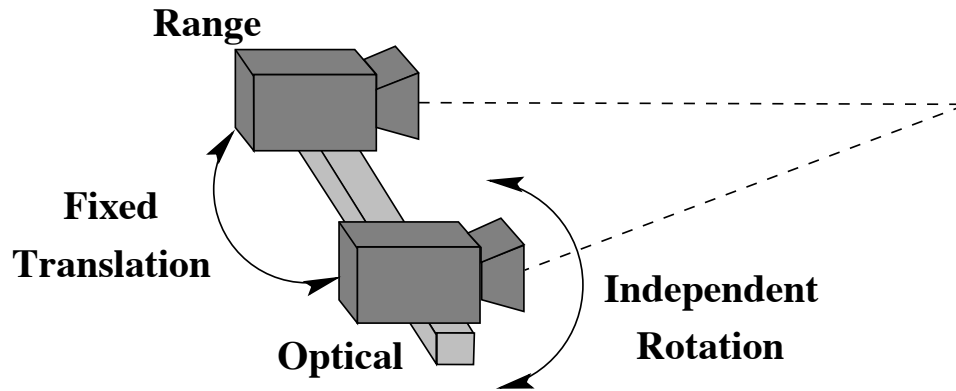


Figure 2.6: Fixed Translation of Sensors, Slight Rotations

The Co-Registration process begins with the initial position hypothesis determined from the 2D range template matching of Phase II. Model features are then extracted from a model database and used in an eight degree of freedom error reduction to determine the ideal transformations [SB94]. Two sets of model features need to be matched to the sensor data: one set for the 3D range data, and one for the 2D optical imagery. Algorithms to extract both pieces have been developed and will be presented in Chapter 5. The next section discusses why these two features were chosen. Chapter 4 presents several data triplets (Color, FLIR and LADAR), and further discusses the sensor characteristics.

### 2.4.1 3D Silhouette

The vehicles being sought in the optical sensor data span only a small number of pixels in relation to the size of the overall image. A cursory estimate places the number of optical pixels on target at roughly two percent of the total number of image pixels present. The coarse level of information present raises two important issues: how much information is present, and how much can automatically be extracted? The answer to the first question defines the information available to work with, and the second defines the subset of usable information.

Since most of the detail from the optical sensor is coarse, we need to determine the most stable optical data features to use in the matching process. Military vehicles typically blend in well with their surroundings. However, vehicle camouflage can not perfectly match all possible backgrounds. The mostly likely place the camouflage will break down in the optical imagery is on the boundary between the vehicle and boundary pixels. This boundary is referred to as the vehicle silhouette. Figure 2.7 shows a color image with the model silhouette rendered with red lines. It is important to note the red lines represent the 3D model silhouette, rendered onto a 2D image. The 3D line endpoints are still known. A full discussion of the generation of the silhouette appears in Chapter 5.

By extracting only the silhouette of the object from both the optical imagery and the model database, internal (within the boundary) vehicle information is lost. In some cases, loss of internal structure is regrettable, since some internal features are typically visible and distinctive. However, choice of the silhouette may be thought of as a first, rough heuristic for determining what features are likely to be visible. A better but more involved approach would be to use lighting model and shape analysis to better predict visibility. Future work will pursue this line of reasoning.

Figure 2.8 shows several views of a model silhouette, generated by the algorithm presented in Chapter 6. The images show the silhouette lines (in red) from a variety of viewpoints. The lines in blue were initially believed to be on the silhouette, but were discarded by later processing. The silhouette was generated for the model orientation shown in Figure 2.8e, and corresponds to the orientation for the image shown in Figure 2.7.



Figure 2.7: Model Silhouette Shown with Color Image (See Color Plate 5)

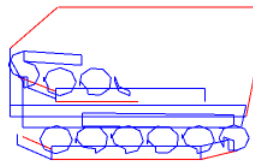
#### 2.4.2 3D Sampled Surface

The range sensor data provides a key piece of information about the scene not otherwise attainable in a 2D optical image: depth. Thus the 3D nature of the range data requires different features be extract from the model than were needed for the optical imagery (i.e. silhouette features). Since the range image contains a set of 3D points sampled from the vehicle, the most comparable model representation is a similar sampling of 3D points. Thus, we need to extract a similar set of sampled surface points from the model for matching. A ray casting algorithm is used to mimic the LADAR sensor and produce a range image similar to the actual sensor. The corresponding features from the data will be extracted based on the region of interest determined by Phase II of the ATR algorithm. By using outlier detection, points in the range image which do not have a corresponding point from the model will be eliminated.

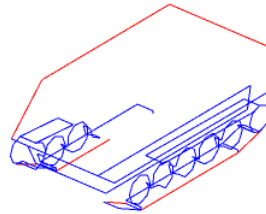
A LADAR sensor image is shown in Figure 2.9a. A M113 Armored Personnel Carrier (APC) model has been rendered with the data in Figures 2.9a and 2.9b <sup>2</sup> in an effort

---

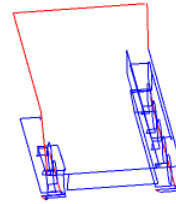
<sup>2</sup>Some points may be obscured by the model



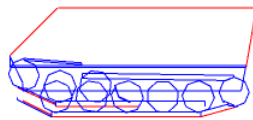
**a.** Az: 180 El: 30



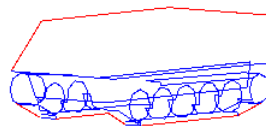
**b.** Az: 229 El: 30



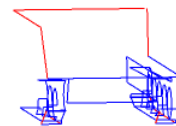
**c.** Az: 278 El: 30



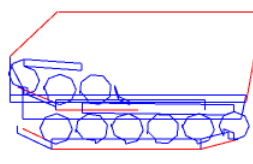
**d.** Az: 180 El: 5



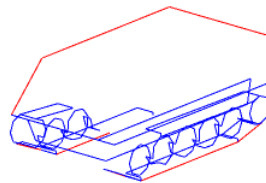
**e.** Az: 229 El: 5



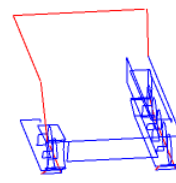
**f.** Az: 278 El: 5



**g.** Az: 180 El: -20

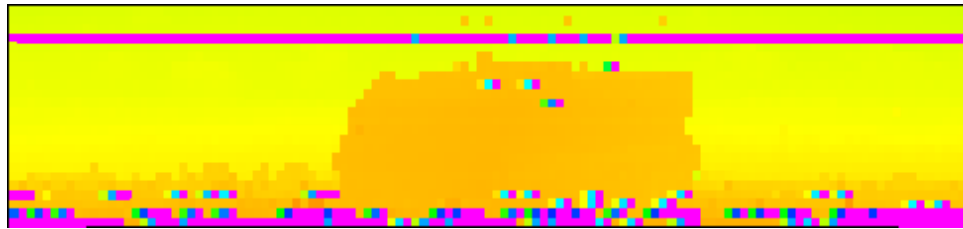


**h.** Az: 229 El: -20

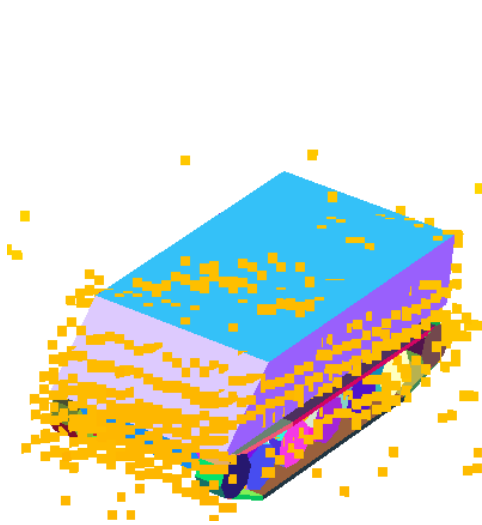


**i.** Az: 278 El: -20

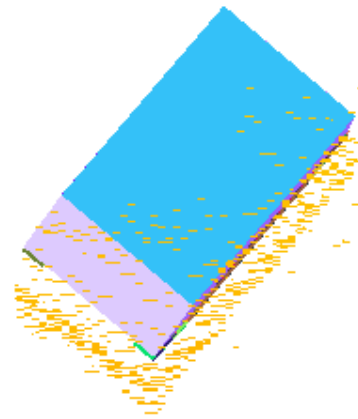
Figure 2.8: Silhouette from Various Angles (See Color Plate 7)



a. LADAR Image



b. LADAR with Model



c. LADAR with Model

Figure 2.9: LADAR Imagery (See Color Plate 6)



to show the relationship between the 3D range data and the model. The results of the technique for extracting the sampled surface from the model as described in Chapter 6 are shown in Figure 2.10. The same surface is shown from a variety of viewing orientations, with the correct viewing orientation (similar to that of Figures 2.9a, 2.9b and 2.9c) shown in Figure 2.10e. The coloring of the range pixels is based on the color of the face which generated the pixel <sup>3</sup>.

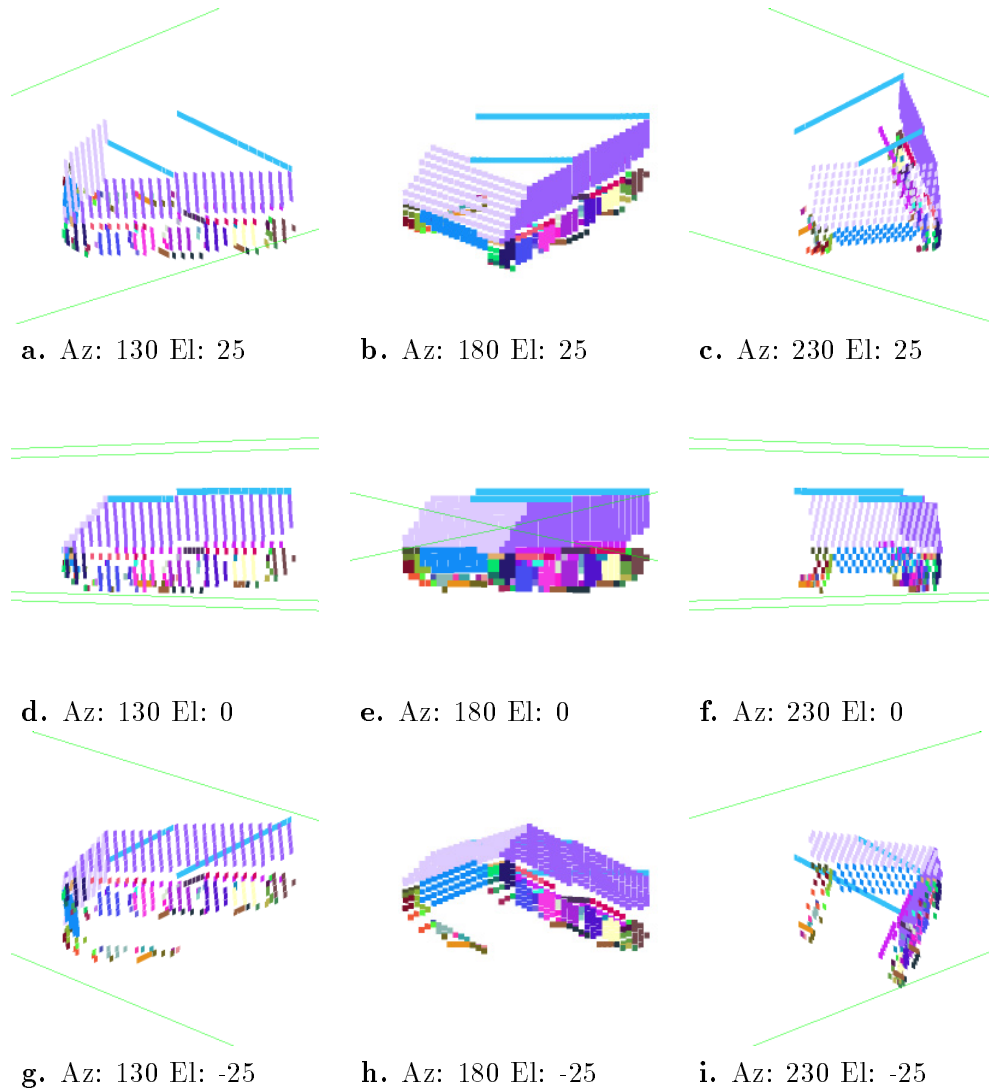


Figure 2.10: Sampled Surface from Various Angles (See Color Plate 8)

---

<sup>3</sup>See Chapter 6 for further details

## Chapter 3

### RANGEVIEW: VISUALIZATION AND VERIFICATION OF MULTI-SENSOR OBJECT RECOGNITION

#### 3.1 Overview

RangeView was the precursor to the system developed for this thesis. RangeView is a tool which allows both the visualization and verification of the results of the Co-Registration process. The tool allowed for the hand selection of features from both the model and sensor imagery for use in the matching process. Once the tool was built it became obvious further work was needed to automate the model feature extraction process. Even though the extraction portion of this tool is obsolete, it did provide the foundation for the work presented here. Even though the verification portion of RangeView is obsolete, it does contain a more powerful visualization engine than the tools developed for this thesis.

#### 3.2 RangeView: A Tool for Multi-Sensor Visualization

After the initial data collection in November 1994, it became obvious that there did not exist a standard tool for visualizing data range, color and IR data simultaneously. Furthermore, we lacked a tool for visualizing the ATR process. Since verification of features and objects recognized from multiple sensors is a non-trivial task [AN90], we needed to develop a tool which would allow us to visualize the relationships between the various sensors, and how they related to our 3D models. With these goals in mind we developed a system known as *Range View* [GBSF94, GBSF95].

Previous systems[RTKM89, VDL94] locate the 3D target in a range image, and then render the model into a 2D scene along with the data. A 2D image neither allows a

complete understanding of how well the target has been located, nor provides a detailed understanding of how the 3D range and model features relate. With our verification system, the 3D model and sensor data can be interactively examined to assess the quality of the match. We have found the ability to arbitrarily change viewing parameters invaluable in the development of our Co-Registration and object recognition algorithms.

### 3.3 RangeView Environment

Our visualization environment, RangeView, combines range imagery, color imagery, thermal (infrared) imagery, and the converted BRL/CAD models of objects being recognized. Several control panels are present which allow the user to view 2D images of both optical sensors. The user has the ability to zoom, crop, and pan in each of the FLIR and Color viewers. The range viewer is the main viewer which allows the separate images to be viewed simultaneously. This viewer also allows the user to interactively view the data and the reduced BRL/CAD model. A set of rotation and translation tools allows the model to be interactively moved through the scene. The user also has the ability to manually register each of the various sensor images by selecting corresponding points in each image. Shown below in Figure 3.1 is an image of the RangeView screen as well as a key for the different viewers in Figure 3.2 <sup>1</sup>.

In order to visualize multiple sensor data three dimensionally, we begin with the range imagery rendered as a set of 3D polygons. the optical imagery is then mapped onto these 3D polygons. By using the optical imagery as a sort of 3D texture map, the resulting registration output from the vision system can be visualized. Instead of viewing the image with color assigned by depth, the image is viewed with the color as coming from either optical sensor. Figure 3.3b shows the range image with the corresponding color image mapped on to it. Figure 3.3b shows the same image with the FLIR mapped on top. The RangeView visualization environment was used to generate many of the images presented in this thesis.

---

<sup>1</sup>These windows may be placed independently on the screen, and the key is only present to aid in identifying the windows for this particular layout

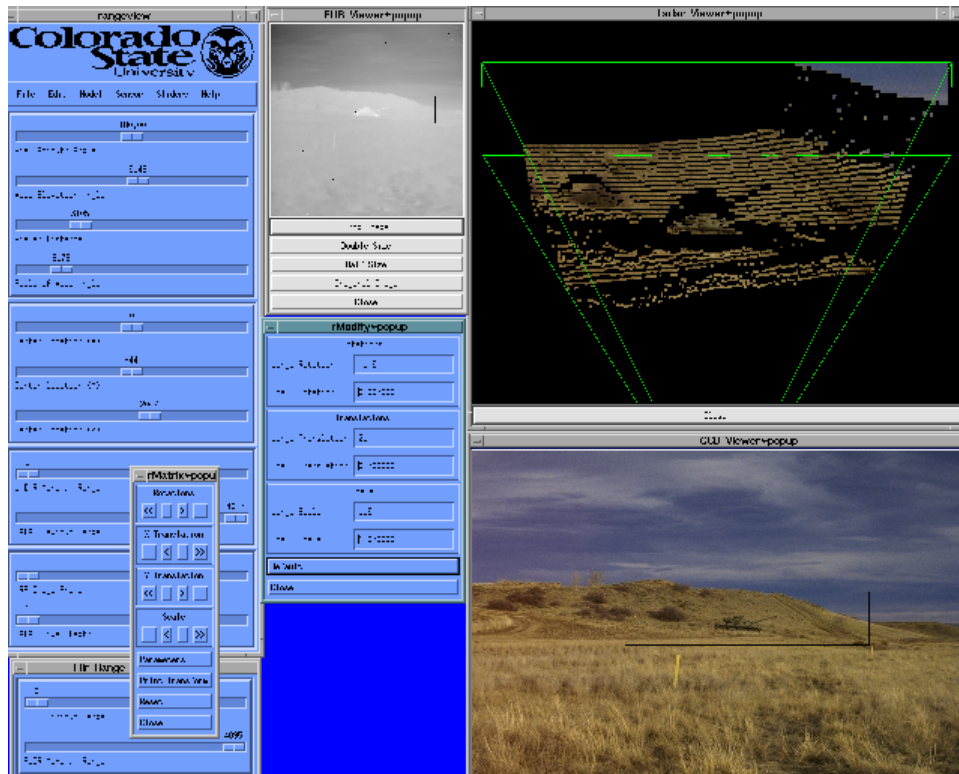


Figure 3.1: The RangeView Screen (See Color Plate 9)

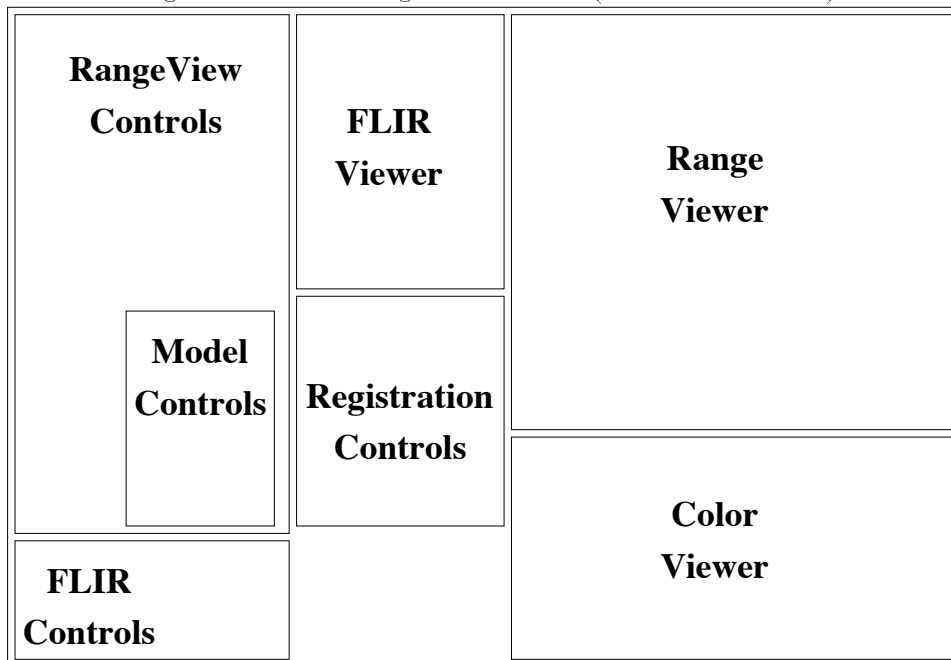
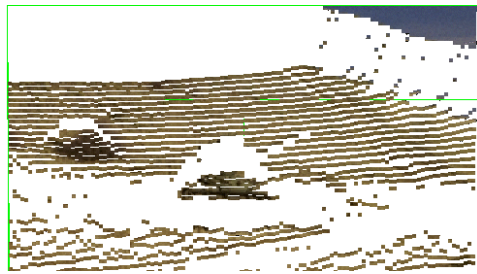
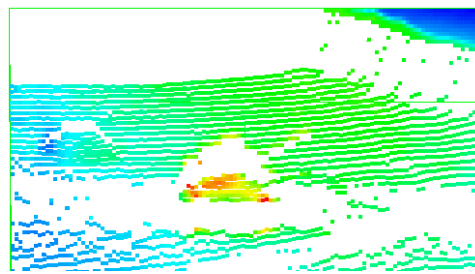


Figure 3.2: Key to understanding the RangeView Screen



**a.** Color and Range



**b.** Thermal and Range

Figure 3.3: Optical Imagery Texture Mapped onto 3D Range Data (See Color Plate 10)

## Chapter 4

### SENSOR DATA

#### 4.1 Overview

The purpose of this chapter is to gain a more thorough understanding of the functionality of each sensor type and how they interrelate. Since the sensor information drives all stages of the model reduction and feature extraction algorithms, it is imperative to understand how each image type relates to the CAD model.

The data set used in the testing of the algorithms developed at Colorado State University was collected in November of 1993 at Fort Carson, Colorado [BPY94]. The Multi-sensor ATR approach taken is desirable because each sensor has its own unique characteristics. By integrating the three sensor types a more stable, robust matching system is expected.

#### 4.2 Sensor Comparison

The FLIR sensor is a thermal sensor which is excellent at locating objects based on their external temperature. The current sensor uses the lower 3-5 micron bands. At the higher 8-12 micron thermal bands, only emitted thermal energy is sensed. However at the 3-5 micron range, emitted energy combines with reflected thermal energy. Thus the effects of solar loading (heat gain due to sunlight) and solar reflectance tend to reduce the effectiveness of the FLIR sensor during the daytime.

The color sensor is a good complement to the FLIR sensor because it is most effective during the day time. While the FLIR is less effective during the day, the color sensor performs best during this time. Conversely the color sensor is worthless at night, whereas

the FLIR sensor can be used to accurately locate a vehicle without the interference of solar reflectance.

The LADAR works equally well at all times, night or day, and it provides a comparison or refinement that could not otherwise be had with only one sensor. The key advantage of the LADAR sensor, as with any range imagery, is it provides a direct measure of 3D geometry. LADAR also has the unique ability to account for occlusion. In a 2D image it is very difficult to determine how vehicle information is lost due to occlusion (either to self occlusion or obstacles in the terrain, or even the terrain itself). However, the LADAR image will show what are believed to be occluded features as large changes in depth. Thus using the multi-sensor approach will allow the recognition algorithm to explain missing features in the optical imagery when LADAR indicates the vehicle is being occluded.

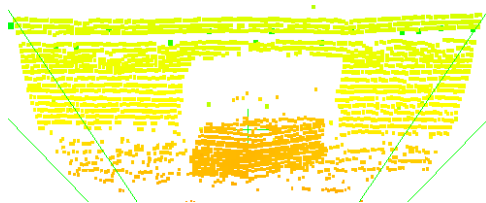
### **4.3 Image Groups**

The three example image sets which will be referred to in both Chapters 6 and 7 are shown in Figures 4.1, 4.2, and 4.3. The three different sensor images are grouped together in each figure to allow an easier comparison of the information each sensor provides. The image sets have also been given a subjective difficulty rating where group-1 is considered easy, group-2 medium difficulty, and group-3 hard. This rating was assigned based on the number of pixels representing the vehicle, the lighting, and orientation of the vehicle. The three sets span the range of difficulty present throughout the entire data set.

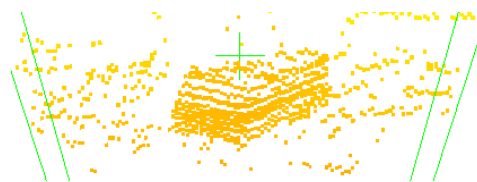
## **4.4 Three Dimensional Range Images - LADAR**

### **4.4.1 Sensor Characteristics**

The Fort Carson data collection incorporated a laser ranging sensor known as LADAR. The LADAR sensor scans a scene in a series of parallel vertical strips, generating a rectangular array of range values with 12 bit resolution. The field of view of the current LADAR system is approximately  $15^\circ$  horizontally and  $3^\circ$  vertically. The maximum range of depth values is approximately 300m. The sensor used was approximately six years old and provides a one foot per pixel on target sample resolution at 150m.



a. Azimuth: 180.0, Elevation: 6.79



b. Azimuth: 180.0, Elevation: 18.34



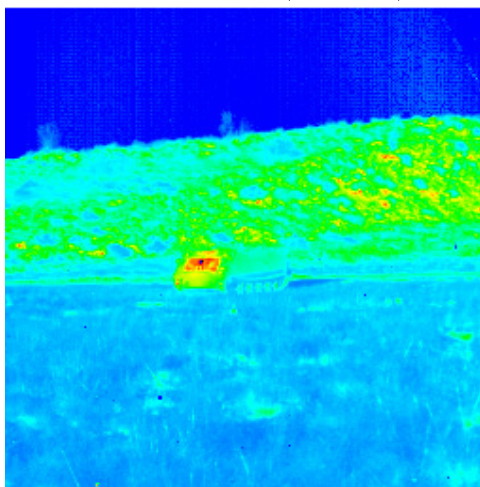
c. Straight ahead LADAR Image



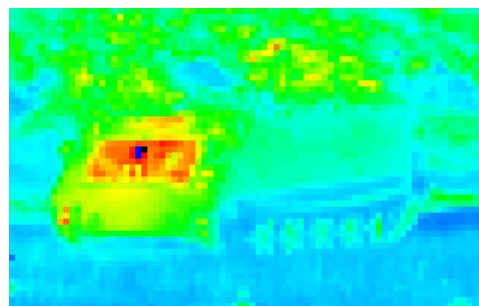
d. Full Color Image (720x480)



e. Corresponding Enlargement



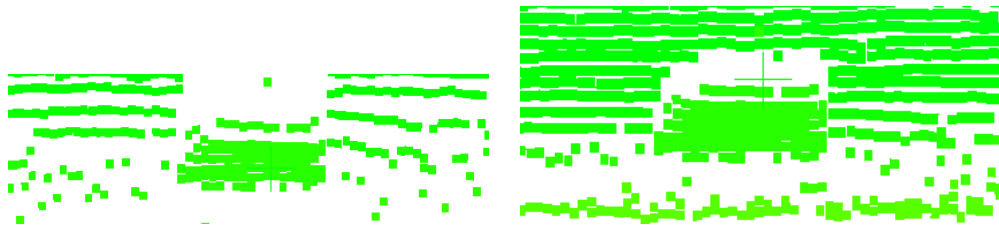
f. Full FLIR Image (256x256)



g. Corresponding Enlargement

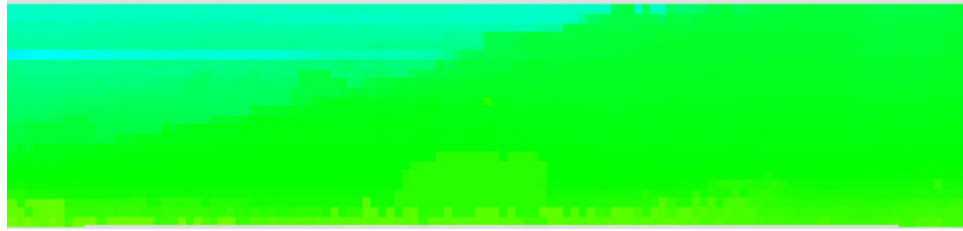
Figure 4.1: Imagery Group Number 1 of M113 APC (See Color Plate 11)





**a.** Azimuth: 180.0, Elevation: 6.79

**b.** Azimuth: 180.0, Elevation: 18.34



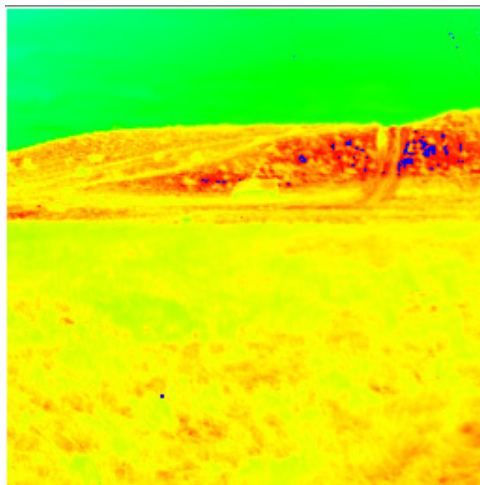
**c.** Straight ahead LADAR Image



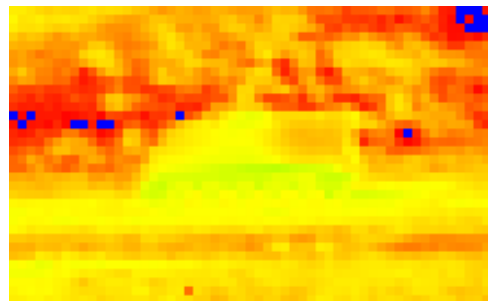
**d.** Full Color Image (720x480)



**e.** Corresponding Enlargement

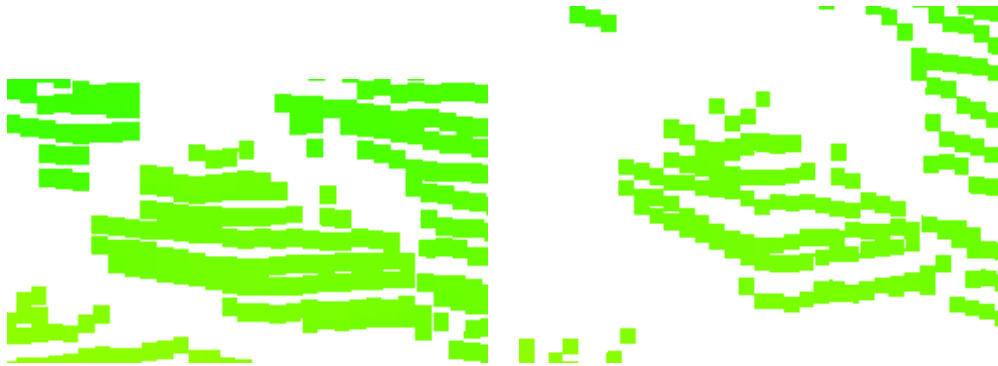


**f.** Full FLIR Image (256x256)



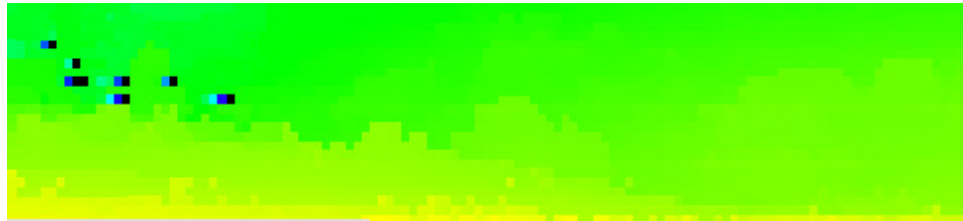
**g.** Corresponding Enlargement

Figure 4.2: Imagery Group Number 2 of M113 APC (See Color Plate 12)



**a.** Azimuth: 180.0, Elevation: 6.79

**b.** Azimuth: 180.0, Elevation: 18.34



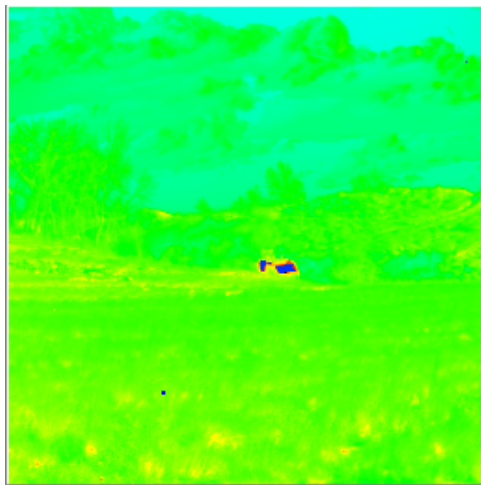
**c.** Straight ahead LADAR Images



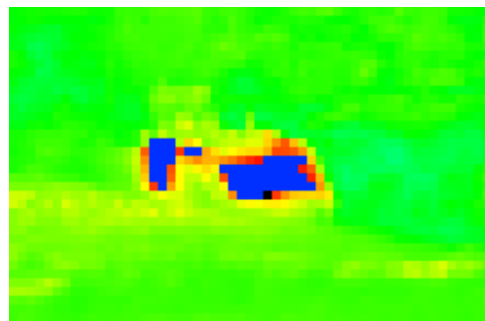
**d.** Full Color Image (720x480)



**e.** Corresponding Enlargement



**f.** Full FLIR Image (256x256)



**g.** Corresponding Enlargement

Figure 4.3: Imagery Group Number 3 of M113 APC (See Color Plate 13)

This LADAR sensor does not accurately reflect the current state of technology in the laser-ranging field. The RSTA project is in the process of acquiring a new LADAR device, called the Schwartz LADAR, which is more sensitive, and provides a 0.3m per pixel on-target sample resolution at 1 km. With this device, developed by SEO inc., there will be similar number of pixels on target to the device used in Fort Carson Data collection, but at 1,000 m rather than 150m. The other sensors will also be affected by the change in distance.

Several range images are shown in Figures 4.1c, 4.2c and 4.3c. The images in each figure represents a “straight-ahead” look, and the two images above represent slight changes in elevation in order to gain a better understanding of the data. All the images were generated from within RangeView. The images are all of the same vehicle type but from different orientations and positions.

These images highlight the 3D information LADAR provides. Note the level of detail present. The first image group contains many pixels on target because the vehicle was very close to the sensor (50m). The image provides an excellent example of the characteristics of the sensor, but can be misleading because the vehicle is so close. In the ATR domain, such high resolution is not realistic and therefore the second and third image sets are more representative. However, in the second image set there is very little information about the target, since most of the samples are planar (equidistant from the sensor). In the third image, the appearance of the vehicle is almost indistinguishable from the appearance of a tree or large boulder in the data.

These very different image sets were used to estimate the level of detail that was needed by the feature extraction and model reduction algorithms. Figure 4.1c, contains many vehicle pixels, whereas Figure 4.3c does not. The model used for matching must meet the level of detail required by all three images. These images were therefore used to guide the reduction phase discussed in Chapter 5.

#### **4.4.2 Calibration**

The exact geometric properties of the LADAR sensor must be known if the sampled surface technique presented in Chapter 6 is to match the actual sensor. Therefore, this

section records this information for the LADAR used at the Fort Carson data collection [BPY94]. A LADAR device consists of a set of vertically aligned sensor units which read the return result of a laser beam for a given direction. Each vertical sensor unit fires a beam which bounces off a mirror and into the scene, and as the mirror is swept left to right, an array of return depths can be created.

The LADAR used in the data collection was calibrated prior to shipping to Fort Carson. In order to field calibrate the LADAR device, two different tests were performed: the first involved aiming the sensor at a telephone pole (for vertical field-of-view (FOV) calibration), and the second aimed the sensor at a set of railroad tracks (for horizontal FOV).

The first calibration test is based on the geometry shown in Figure 4.4a. The Figure shows each of the vertical sensor units labelled with a number. Sensor 1 is the lowest unit in the array, and Sensor 10 was the sensor to return the correct distance to the telephone pole. Assuming the laser range is known to be 157 Ft, and the vertical distance between the sampled points is known to be 3.7 Ft, we can calculate the vertical field of view based on the following equation:

$$V_{fov} = \tan^{-1}\left(\frac{R}{(D)(NumSensorUnits)}\right) \quad (4.1)$$

where  $D$  represents range to target,  $R$  the vertical distance between samples, and  $NumSensorUnits = CorrectDistanceSensorUnit - LowestSensorUnit = 10 - 1 = 9$ . Using this equation we can obtain the measured result of  $2.61mRad$ . Based on the actual sensor characteristics a value of  $2.639mRad$  was predicted, which produces a range accuracy of 1 percent <sup>1</sup>.

The second calibration test was based on the geometry shown in Figure 4.4b, and was used to determine the horizontal FOV. The Figure shows the response of the top vertical sensor unit as the mirror is swept across the scene. Again the first unit's response

---

<sup>1</sup>Accuracy is measured as  $\left(\frac{Predicted-Actual}{Predicted}\right)$

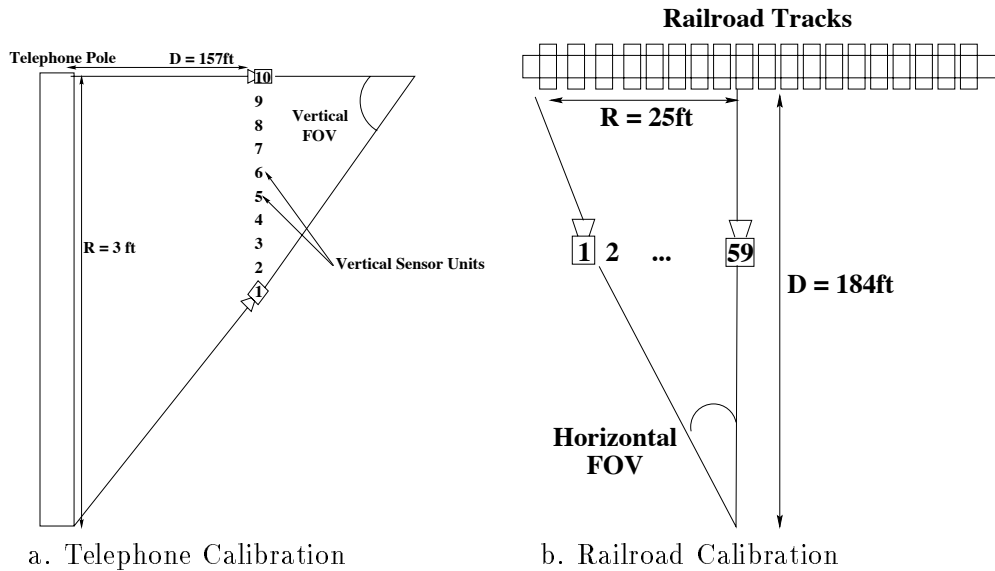


Figure 4.4: Calibration Geometry for LADAR Device

was recorded as well as the first sensor unit to return the correct distance to the railroad tracks. The horizontal spacing between sampled points was determined to be 25 Ft, and the distance to the railroad tracks was 184 Ft. A modified version of the vertical FOV equation can be used to determine the horizontal FOV. A similar relative error of 1 percent was also determined for  $H_{fov}$ .

## 4.5 Color Images - 35 mm

### 4.5.1 Sensor Characteristics

The color imagery was obtained with a standard 35mm camera. The images were digitized to a Kodak Photo Compact Disk. The process produced high quality scene images. The images have a dimension of 720x480 pixels, and are stored in the tiff 24-bit image file format.

The Color images corresponding to the LADAR images shown in Figures 4.1c, 4.2c, and 4.3c are shown in Figures 4.1d, 4.2d, and 4.3d. Again, it is important to notice the level of detail present in each image set. The color images in Figure 4.1d, have excellent on target pixel resolution, but Figures 4.2d and 4.3d do not. It should also be noted that while both vehicles blend in with their surroundings, there does exist a silhouette boundary between vehicle and non-vehicle pixels. The silhouette from the reduced model can be extracted at the level of detail present in both Figures 4.1d and 4.2d.

### **4.5.2 Calibration**

Images of a geometric test pattern were shot each day at Fort Carson. From these images it is possible to derive the focal length, image center, horizontal and vertical field of view. A full discussion of the calibration algorithm is presented in [BHP94] and summarized below.

Three images are taken of the color test chart, the Macbeth Checker Board, each at different distances from the sensor. A set of points is then extracted from each of the three images, and determines the actual 3D coordinate which corresponds to that pixel. Both the 2D corresponding points and the actual 3D points are fed into the equations derived in [BHP94], in order to determine the actual sensor parameters.

## **4.6 Forward Looking Infrared Images - FLIR**

### **4.6.1 Sensor Characteristics**

The FLIR sensor is an amber FLIR in the 3 to 5 micron range. It is considered a high-end first generation FLIR device and produces 12 bit values representing a heat intensity image. The quality of output is excellent compared to most first generation FLIR sensors. The final images that make the multi-sensor group are shown in Figure 4.1f, 4.2f, and 4.2f. As with the color optical sensor, the silhouette boundary is the most stable image feature that is clearly definable.

### **4.6.2 Calibration**

The optical focal length etc., of the Amber FLIR can be calibrated using the same geometric test pattern which used to calibrate the color video and 35mm cameras. A calibrated thermal source was also available to provide thermal calibration for the FLIR at Fort Carson.

## Chapter 5

### CONVERTING CSG TO POLYHEDRA

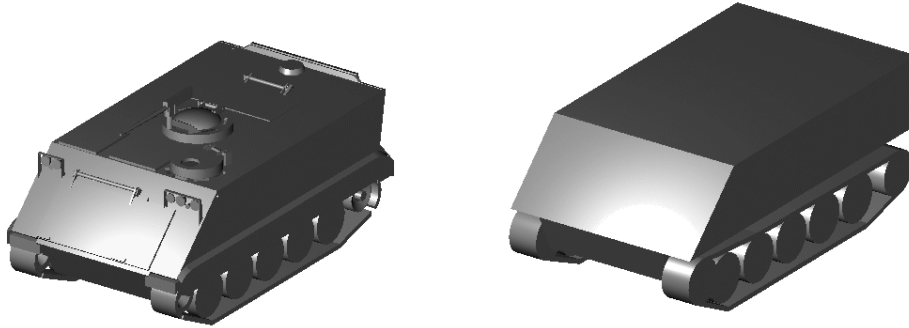
#### 5.1 Overview

According to Snyder [Sny92], polyhedral objects are a useful representation of any type of object that can be bounded by a set of planar faces. Snyder states the two advantages of this type of model are: the ease in which they can be rendered on modern graphics hardware, and the ability to analyze their shape. These two features are important to both our visualization tools (RangeView), and the ATR algorithm (Co-Registration). There are two main problems which need to be solved in order for the current BRL/CAD models to be used by our ATR algorithm: the first is the level of detail present, and the second is the model representation.

The BRL/CAD models were chosen for our use because they contain the most accurate representation of the vehicles for which we have sensor data. However, with their accuracy comes an abundance of information and detail which is below the resolution of the sensor data. The detail in BRL/CAD models is important for structural and vulnerability analysis, which is the primary use of these type of models [Fre95]. However, the models are not meant to reflect all vehicles which could possibly be visible in a scene. Military vehicles are often worn and damaged from heavy use, and specific features and details in the high resolution models may not be present on an actual vehicle. For recognition, what is needed is a model which represents the most generic features of the type of vehicle being sought [SWF95, FH87].

Figure 5.1a is an example of a detailed M113 APC CSG model. Note the level of detail exceeds that of the various sensor images shown in Chapter 4. The reduction of the model to a the level of detail shown in Figure 5.1b, provides the more general M113 APC

shape without the features, such as the headlights and hatches, which are difficult to find in the data and hence unreliable.



a. BRL/CAD Model

b. Reduced Model

Figure 5.1: M113 APC Model

The second problem associated with the BRL/CAD models for our application is the CSG format in which the models originate. The implicit CSG representation of the BRL/CAD models does not lend itself well to readily extracting 3D face and edge information. Converting the models from their initial CSG to polyhedral form yields models in which face and edge information is explicit. The new representation can be readily used to extract the information needed for the ATR algorithm.

## 5.2 Explicit Versus Implicit Model Representations

The motivation for converting the CSG to polyhedra is based on the need for a model representation in which the information needed is explicitly represented. For matching to object silhouettes in optical data, the relevant model representation needs to have 3D edge segments readily available. For the range data, 3D face information needs to be represented. Since CSG representations consist of a set of primitive structures to which boolean operations are applied, significant computation is needed before edge and face information can be readily extracted.

CSG is usually represented as a tree structure where each node represents an object created by applying a set operation to its children. The leaf nodes, or nodes with no



children, are considered the primitives of the modeling system. An example of such a graph can be seen in Figure 5.2a. The conversion algorithm essentially replaces the entire tree with a set of polygonal faces that approximate the object. The resulting polyhedral object for Figure 5.2a is shown in Figure 5.2b.

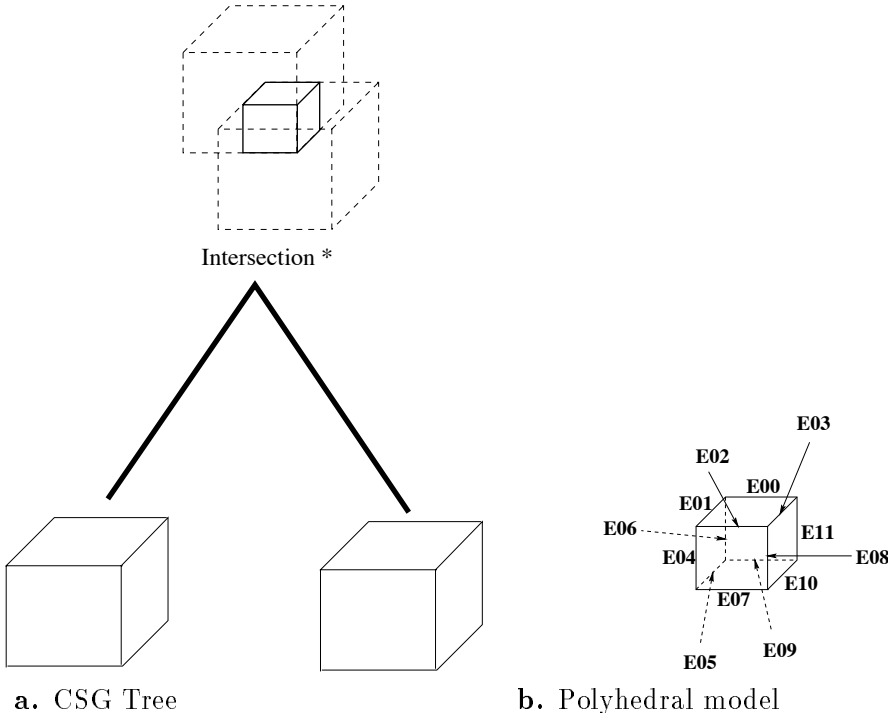


Figure 5.2: Intersection of Two Cubes

Since model matching can be computationally expensive, the model’s representation should closely resemble the information that needs to be extracted [Pop94]. We have chosen to use a Boundary Representation (B-Rep) as our model database [Man90]. Figure 5.3 shows an example for the intersection of the two cubes. Notice the explicit representation of the face and edge information needed by our ATR algorithm.

**5.3 Model Reduction**

When our RSTA project initially began, we spent several months attempting to automate the reduction phase of the algorithm as a separate part from the conversion from CSG to polyhedra. However, we discovered, as others have before us [RV80, LTH86], that the conversion algorithm was susceptible to floating point error associated with converting the mathematical CSG model to the finite polyhedra representation. It became obvious

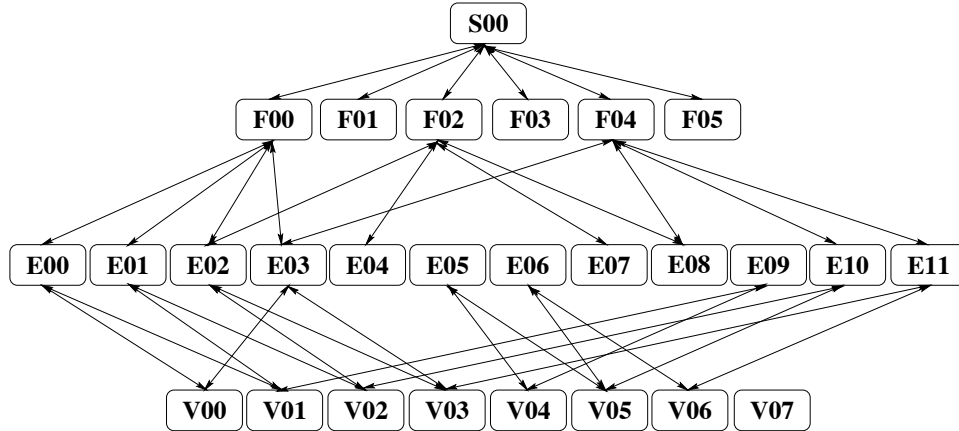


Figure 5.3: The B-rep for the Intersection of two cubes

the reduction phase should be applied before the conversion process in order to increase the accuracy of the converted model (see section 5.4.3.8).

In order to expedite the production of usable models for the feature extraction phase, the models were greatly reduced by hand. All operations were applied within the BRL/CAD interface environment known as MGED (Multi-display Graphical EDitor) [U. 91]. The procedure involved determining how each primitive related to the overall model appearance. Primitives which were determined to be insignificant were then removed. This process allowed features such as hatches, hatch bolts, head lights, and tail lights, to be quickly removed.

The process was also somewhat subjective. Sometimes small details were retained, and other times they were discarded. The process often involved consulting numerous sensor images and those features which could not be distinguished were removed. During this process, no simple, formal rules could be devised which clearly determined why any particular primitive was removed. For this reason the reduction stage was found to be more easily accomplished manually.

In addition to reducing the number of primitives, several modifications to the CSG tree needed to be made. The original models contained several primitives which only intersected at a single point, or resulted in ambiguous solids when the boolean operations were applied (i.e. the union of two cubes meeting at a single point in space results in a point which is not a valid solid [Man90]). The models were edited to remove such

ambiguities. By separating primitives which only overlapped by a few millimeters, the conversion process produced a more accurate model.

## 5.4 Model Conversion

After the BRL/CAD models are reduced to a minimal set of primitives required, they are converted to a polyhedral representation. Several different algorithms were examined [RV85, PS86, NAT90] before the Laidlaw [LTH86] algorithm was chosen for implementation. These methods are all based on the principal that each object used in a boolean operation can be split into a set of planar faces, each of which lies completely inside or outside the other objects. A classification algorithm can then be used to determine which faces are retained and which are discarded to produce the final polyhedral model.

### 5.4.1 Related Research

Solid Object modeling is concerned with the representation and description of mathematical objects. Currently the BRL/CAD models we have contain many different types of convex 3D objects combined together with several set operations. Many 2D algorithms, such as [Vat92] and [WA77], exist for this conversion, but since they do not fit our 3D needs, they were left out of the following discussion.

In 1980, Aristides Requicha and Herbert Voelcker [RV80] presented a description of a modelling format known as Constructive Solid Geometry (CSG). Their format was based on the belief that complex objects could be constructed by applying various set theoretic operations on simpler objects. By applying the set operations of union, intersection and difference to a variety of simple objects, a wide range of complex models could be produced.

In 1983, Requicha and Voelcker [RV83] presented a paper discussing the then current research directions in solid modeling. One of the directions they examined was converting models between different formats. At the time the article was written, many other issues in solid modeling had a higher priority and needed to be conquered before the conversion problem could be solved. Therefore the polygonalization of CSG models was not considered at the cutting edge of modeling research. It was not until 1985 when Requicha

and Voelcker [RV85] presented a paper on performing the conversion task that alternative solutions to the problem began to be considered.

The algorithm proposed by Requicha and Voelcker performs boolean operations on polygonal models by determining all the boundary points between each child node of the CSG tree. Each planar face of an object on the left branch of the tree is intersected with all faces of the object on the right branch. This forms a set of line segments which can be broken into pieces representing one of the following: 1) portions of the object on the line, 2) portions of the object not on the line, and 3) portions of the line shared with the other object. Based on these classifications, and the required set operation, the object can be converted to a set of polygons representing each node on the tree. This operation is applied recursively until the entire tree has been converted.

Requicha and Voelcker also recognized a slight problem with the use of purely mathematical set operations. When dealing with solid models, set theoretic operations can sometimes produce what are known as dangling edges or faces. This phenomena can be solved by incorporating regularized set operations instead of the strictly mathematical union, intersection and difference. These three new operations, referred to as union\*, intersection\*, difference\*, reject dangling edges when applied to solids. See Figure 5.4 for a description of a dangling edge, and the regularized union operation. Requicha's and Voelcker's algorithm worked with these new operations that are now referred to as regularized-set (r-set) operations.

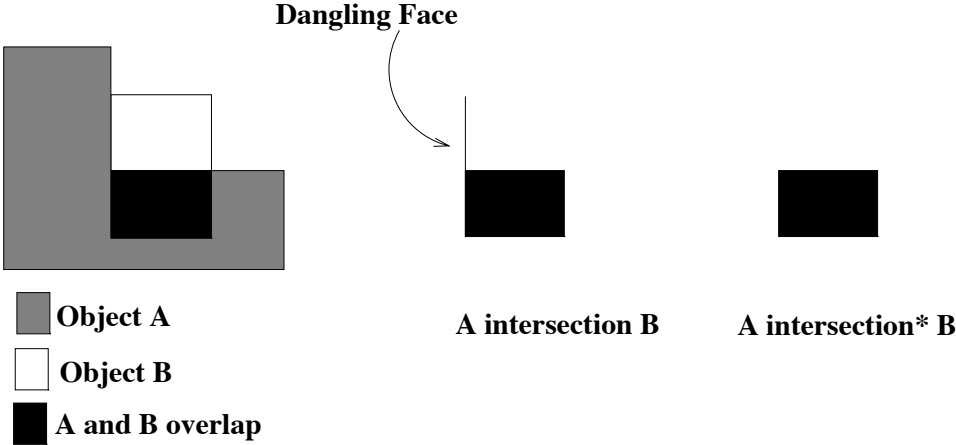


Figure 5.4: Regularized Set Operations

In 1986, David Laidlaw [LTH86] developed an algorithm based on Requicha's and Voelcker's work. The algorithm presented in [RV85], left several implementation details unspecified. Laidlaw addressed these issues, and presented a complete working algorithm to accomplish the conversion task.

Like the Requicha algorithm, the Laidlaw algorithm applies the r-set operations to every possible pair of objects in the CSG tree. For each pair of objects, line segments similar to that described by Requicha and Voelcker are obtained. A segment is used to split a face into a set of convex faces which are completely contained within the other object or completely outside of it. A classification routine then labels each new face and based on the result, and the face is added to the resulting object or it is discarded. Since the only requirements for the algorithm to work properly are a set of convex, planar faces grouped to represent an object, the algorithm lends itself well to our model conversion problem. The algorithm also has some additional advantages that will be discussed later.

In 1986, L. K. Putnam [PS86] proposed another method to perform boolean operations on n-dimensional objects. The algorithm is similar to Laidlaw's approach. The algorithm first determines all the boundaries between two objects. Each edge of the object is then split at that boundary and labelled. The labelling produces edges that are classified as being part of the produced object, or not. After all the labelling and subdivisions have taken place, only the edges with labels matching the requirements of the boolean operation are retained. The main difference between this algorithm and Laidlaw's approach is the representation of the initial CSG model, and its reliance on edge information instead of Laidlaw's face driven approach. We chose to use the Laidlaw algorithm because its data structures are better suited to the original BRL/CAD model format.

In 1990, Bruce Naylor [NAT90] described a different algorithm to accomplish the same conversion. Naylor's algorithm first generates a Binary Space Partition (BSP) tree for each object to be used in the process. A BSP tree is a hierarchical set of polygons with each parent and child being separated by a hyper-plane, or some division of Euclidean space which completely separates both nodes. Naylor presents a method, based on the required theoretic set operation, which merges the two BSP trees. The algorithm continues until all the objects in the model have been processed.

The BRL/CAD primitives could be converted to BSP trees and then Naylor’s algorithm could be applied, but the mapping between representations is not as intuitive as the mapping to the representations used by Laidlaw. For our implementation, Naylor’s algorithm has the added overhead of converting from the BRL/CAD format to the BSP trees, and then back to the polygonal representation used by the ATR algorithm. In addition, Naylor’s algorithm is extremely susceptible to numerical errors associated with finite arithmetic. Laidlaw’s algorithm appears to be more numerically robust for our specific application domain.

Many of the algorithms that exist to convert CSG models to a polygonal representation first convert the CSG to another representational form before the set operations are applied. The Laidlaw algorithm has the advantage that the BRL/CAD models are already in a form that can be used, and no conversion overhead needs to be introduced. The algorithm was also developed to be more numerically robust than some of the other approaches examined. These benefits, coupled with its recursive nature (the result of a boolean operation can be used in another operation), supported the choice of this algorithm.

#### **5.4.2 BRL/CAD Models**

The BRL/CAD models we are using contain several different primitives that are grouped together along with the three main set theoretic operations (union, intersection, difference) in a CSG format. Currently, our algorithm only supports a subset of the primitives available in BRL/CAD. The primitives supported are: five types of arbitrary convex polyhedron, right circular cylinder, and an interpolated curve. The representation of each format is formally described in the BRL/CAD user’s manual [U. 91]. Figure 5.5 shows the various primitives as depicted by MGED. BRL/CAD currently supports upwards of twenty seven different primitives, and more are added with each release. The primitives chosen for this conversion algorithm are those present in our models. However, adding new primitives to the conversion process is a trivial task provided a polygonal representation of the primitive exists.

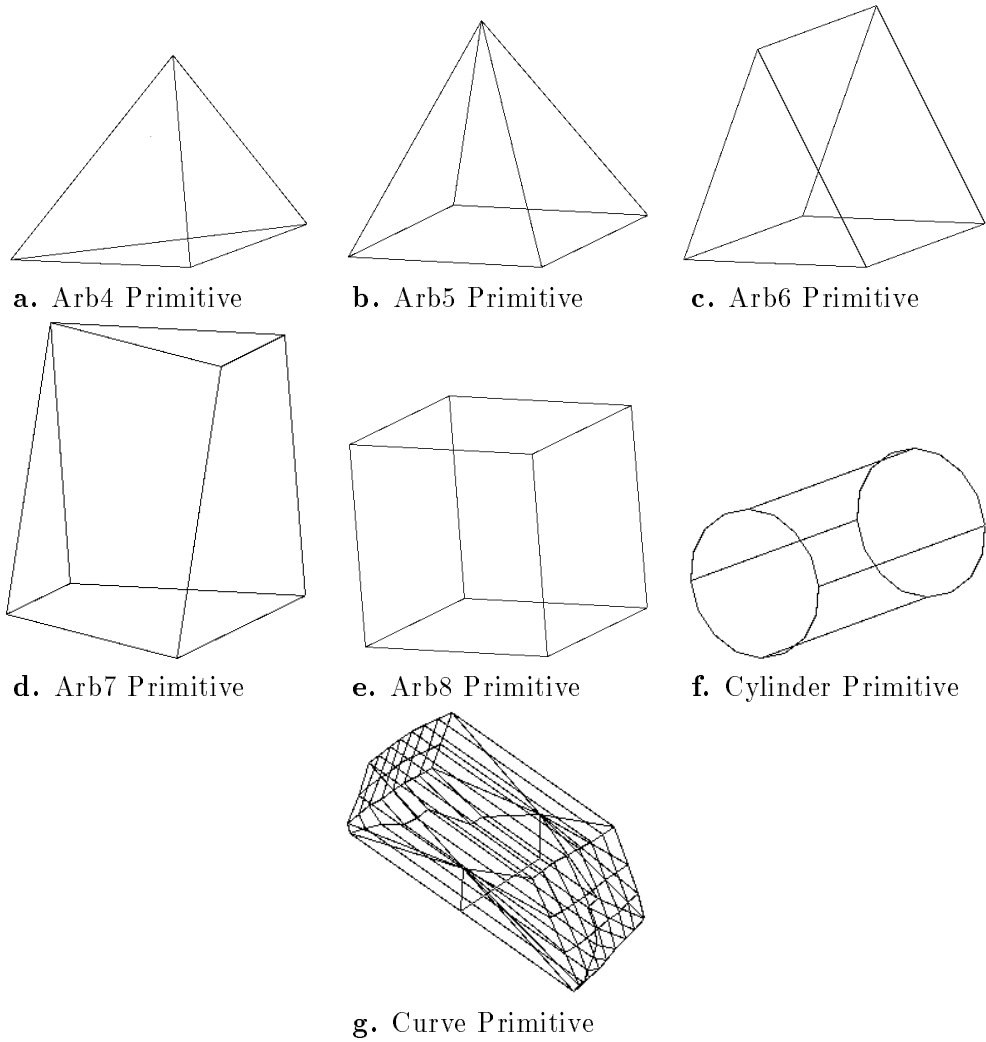


Figure 5.5: BRL/CAD Primitives Supported

The arbitrary convex primitives supported consist of six, five, four, and three sided objects. The BRL/CAD format defines each of these primitives as a set of adjacent faces that group to form a solid object. Since the primitives, shown in Figures 5.5a through 5.5e, are already in a polyhedral form, no conversion is necessary besides the direct translation between the BRL/CAD data structure and our representation.

The conversion algorithm developed also supports the polygonalization of a right circular cylinder, shown in Figure 5.5f. Since the cylinder is represented as a mathematical object, the conversion to a polyhedron will introduce a certain amount of error. The polygonal cylinder is represented with octahedra at both ends, and eight rectangles connecting them together along the sides (see Figure 5.6a). Each face of this object is considered to be planar, with a normal vector pointing in the outward direction. In order to increase accuracy between the polygonal representation and the cylinder, the ends of the cylinder could be represented by polygons with additional points (more points than eight). However, the octahedron is the most common representation [FvDFH90].

The final BRL/CAD primitive supported is an interpolated curve. Figure 5.5g depicts a BRL/CAD view of a portion of an M60 tank turret. The BRL/CAD primitive contains several such curves that are grouped together to form a solid. To convert these curves to polyhedra, several faces are needed. The first two represent the ends of the solid, and are obtained by using the curve control points as the vertices of the face. Rectangles are then created to connect the ends together and form the solid object, see Figure 5.6b.

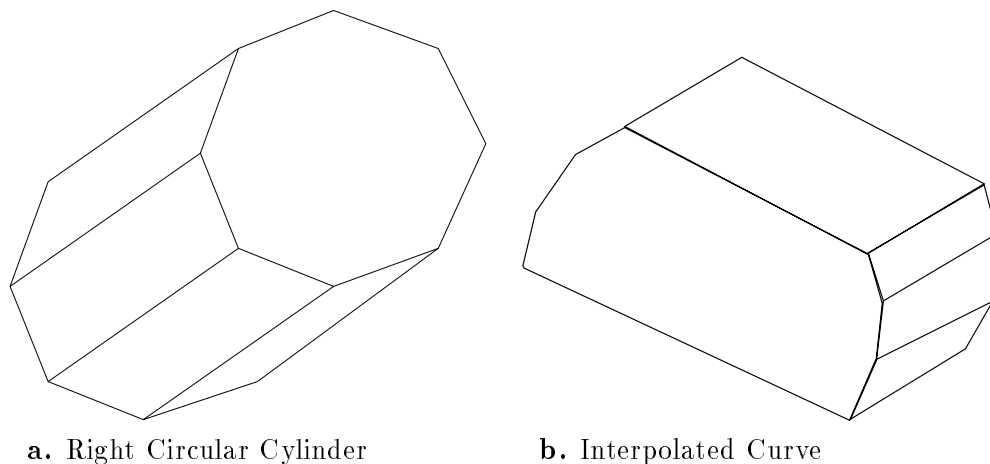


Figure 5.6: Converted Primitives



The error introduced by the conversion process is presumed to be small with respect to the splitting of the CSG objects. In addition, the errors introduced will fall below the accuracy of the sensor. If the sensor data is only accurate to one half a meter, conversion errors in the model of a few millimeters are inconsequential.

### 5.4.3 The Conversion Algorithm

For the reasons stated above, we have chosen to utilize the algorithm developed by David Laidlaw [LTH86] for convert the CSG representation into polyhedra. The conversion algorithm is summarized below, along with some minor refinements which we found improved the quality of the resulting model. For a complete description of the algorithm the reader is referred to [LTH86], or [RV85] which provides a discussion of the mathematical basis for the algorithm.

The conversion algorithm traverses the CSG tree, transforming the leaf nodes from BRL/CAD primitives into polyhedra. Next, each node in the tree is visited with an in-order traversal. The two children of each node are merged into one object by applying the appropriate regularized set (R-set) operation. The R-set operation will generate a polyhedral operation which can then be merged with other objects in the tree.

The first phase of the R-set operations begin by subdividing an object so that none of its faces intersect any face of the other object. After the splitting phase has occurred, a classification process labels each face of each object. The classification routine casts a ray from the center of each face in the direction of the face normal. Based on the intersection of the ray with all the faces in the other object, the face is labelled as either INSIDE or OUTSIDE.

The final phase uses a table look-up to determine which faces of each object are kept, based on the label and the appropriate R-set operation from the CSG tree. The end result is a new solid object which can be used in subsequent operations. The process continues until the entire CSG tree has been traversed and the model has been converted to its new polyhedral representation.

#### 5.4.3.1 Phase I: Subdividing Objects

The algorithm begins by examining two objects which intersect, and determining how to split them. Splitting each object requires three passes: ObjectA is split so that it does not intersect ObjectB, ObjectB is split so that it does not intersect ObjectA. During both of these processes, new edge segments may be introduced. It is therefore necessary to split ObjectA against ObjectB again, thus resolving any intersections introduced from the previous splits.

The pseudo-code for the split process is shown in Table 5.1. In order to limit the number of comparisons needed, and to neglect performing operations on objects that do not intersect, the extents, or bounding box, of objects and polygons (represent part of a face on an object) are compared. If the extents do not overlap, or intersect, computation on the two polygons or objects is ended. Through a series of tests, the algorithm finds two polygons that need to be split with respect to each other.

We have added a slight modification to this part of the algorithm in order to facilitate model reduction. As the two objects are compared, if one of the object's bounding box size is below some user specified percentage of the other object, no splitting occurs. This user parameter is obviously model dependent, but for our application we found it reduces time spent in the splitting portion of the algorithm for situations in which the result of the operation would have produced small fragmented triangles.

If the extents of PolygonA overlaps the extent of PolygonB, then the signed distance test is performed. This test computes the distance from each vertex in PolygonA to the Plane of PolygonB. If all of the distances are zero, the two polygons are co-planar, and if all of the distances are of the same sign, the polygons do not intersect. If the polygons are not coplanar and they do intersect, the line of intersection of the two planes is calculated. The portion of the line formed is called a segment. If the segments of PolygonA and PolygonB overlap, PolygonA is split against PolygonB and the resulting polygons replace PolygonA in the Object. Once the segments are determined, the way in which they cross each of the polygons can be analyzed.

There are only six possible ways a line can cross a convex polygon, and they are shown in Figure 5.7. Once the appropriate crossing type has been found, the start, end

```

if (OverlapExtents(ObjectA, ObjectB))
  foreach (PolygonA in ObjectA)
    if (OverlapExtents(PolygonA, ObjectB))
      foreach (PolygonB in ObjectB)
        if (OverlapExtents(PolygonA, PolygonB))
          DetermineSignedDistance(PolygonA, Plane of PolygonB)
          DetermineSignedDistance(PolygonB, Plane of PolygonA)
          if (Polygons not Coplanar and do Intersect)
            IntersectPlanes(Line, Planes of both Polygons)
            DetermineSegment(Segment of A, Line, PolygonA)
            DetermineSegment(Segment of B, Line, PolygonB)
            if (SegmentsOverlap(Segment of A, Segment of B))
              NewPolygon = Split(PolygonA by PolygonB)
              ReplacePolygon(PolygonA, NewPolygon)
            endif
          endif
        endif
      endforeach
    endif
  endforeach
endif

```

Table 5.1: Splitting Two Objects

and middle points of are determined. This information, coupled with the portion of the line overlapping the polygon can be used to create a segment for each of the two polygons. If the start of the segment for PolygonA (SegmentA) lies in between the start and end of the segment for PolygonB (SegmentB), the starting point of SegmentA is given the endpoint of SegmentB, and its type becomes the same as its own middle type. The process is also performed for the end of SegmentA. If the end of the SegmentA lies in between the start and end of the SegmentB, the endpoint of SegmentA is changed to the start of SegmentB and its type becomes the middle type of SegmentA.

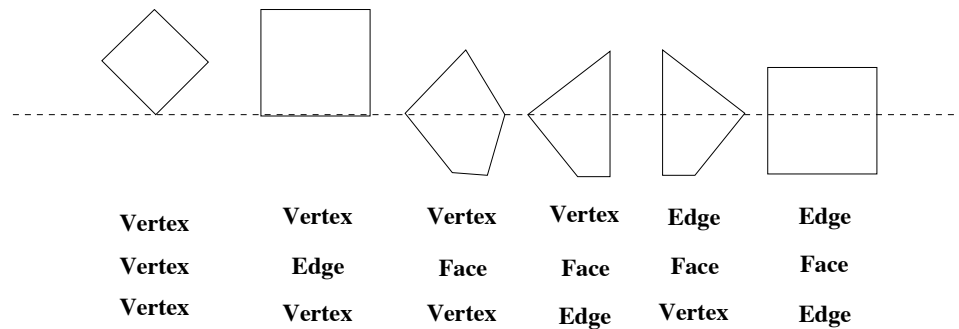


Figure 5.7: Segments formed by a line

The overlapping portion of SegmentA is then used to split PolygonA into a set of convex polygons. There are at most twenty seven possible combinations of types for SegmentA. Of these twenty seven, thirteen are impossible and four are symmetrical cases. The remaining ten cases are shown in Figure 5.8. The classification types are abbreviated above the arbitrary polygons as follows: (V)ertex, (E)dge, (F)ace. B represents the vertex closest to the beginning of the segment (as determined by moving in a clockwise motion around the face), and E is the vertex closest to the end of the segment (as determined by moving in a clockwise motion around the face). N and M represent new vertices which must be introduced into the polygon. The dotted line represents the line formed from the two polygons intersecting, and grey lines are the new edges introduced by the splitting operation.

After the splitting has occurred, no faces of PolygonA will intersect any of the faces of PolygonB. During the splitting, vertices part of the VVV, VEV, VEE, VFV, VFE, VFF, classifications are marked as boundary vertices. These boundaries will play a key role in the next phase of the algorithm.

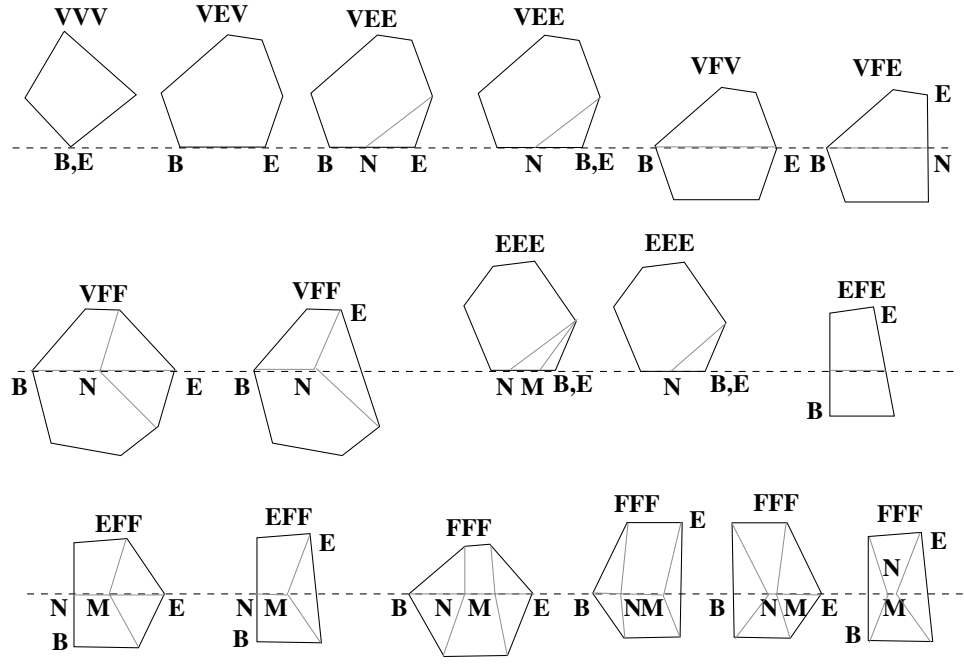


Figure 5.8: Splitting Alphabet

5.4.3.2 A Splitting Example

The intersection of two cubes example of Figure 5.2 requires several splits to three of the faces in ObjectA, and three of the faces in ObjectB. Each face is split in the same pattern: first an EFF split occurs, followed by a VFE. A diagram of this process is shown in Figure 5.9. After the six faces for each cube are split the result is passed on to the labelling phase.

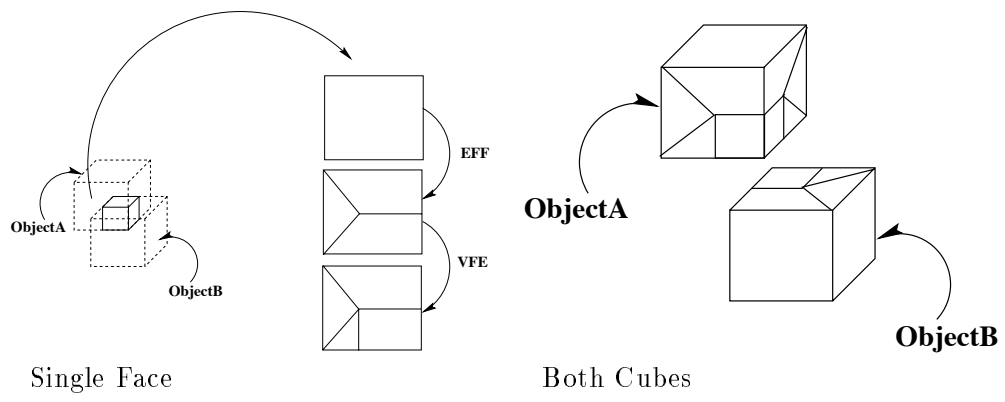


Figure 5.9: A Splitting Example

#### 5.4.3.3 Phase II: Labelling Polygons

The next phase of the conversion process involves labelling each face of the split object as either being `INSIDE` the other object, completely `OUTSIDE` the other object, being the `SAME` face (coplanar, normal in same direction) as a face in the other object, or being the `OPPOSITE` (coplanar, normal in opposite direction) another face in the other object. Since none of the subdivided faces in `ObjectA` will intersect any of the subdivided faces in `ObjectB`, these are the only four possibilities.

The B-rep data structure used to store the various solids represented in the model contains several topological relationships which we can exploit to speed up the labelling process. The B-rep stores which edges share each vertices:  $E(V)$ , as well as the converse relationship:  $V(E)$  (which edges surround each vertex). This vital piece of information, along with the knowledge of which vertices lie on the boundary of both objects, will be used to propagate a known label through the model. The process examines each face, and if any of the vertices surrounding the face do not have a known classification, the ray casting routine is called to determine the label for the face. This label can then be propagated to other adjacent faces through the vertices they share.

The labelling process starts by determining the appropriate label for a given face. Each vertex of the current face are then given this label, provided none of them are already labelled as being on the boundary of the two objects (to which the boolean operation is being applied). Next, each vertex connected to the face through an edge is examined, and the label is passed on. The process continues until a boundary vertex is reached at which time the propagation stops. This process has the effect of labelling several faces without performing the costly ray casting procedure.

The ray casting routine is a time consuming process which examines the desired face (`FaceA`) of an object (`ObjectA`) with respect to all faces of the other object (`ObjectB`). The ray starts at the center of `FaceA`, and is cast in the direction of the face normal. The face (`FaceB`) in `ObjectB` with the closest intersection is used to provide the label. The dot product of the surface normal for `FaceA` and `FaceB`, along with the distance from the center of `FaceA` to the plane of `FaceB`, can be used to provide the four classifications

(INSIDE, OUTSIDE, SAME, OPPOSITE). The pseudo-code for this operation is given in Table 5.2.

```

create ray
foreach FaceB in ObjectB
  find DotProduct of normal for FaceB and ray direction
  find Distance from center of FaceA to FaceB
  if (Distance < 0)
    then no intersection
  if (Distance = 0) AND (DotProduct = 0)
    then ray lies in plane of FaceB,
      perturb ray direction and recast
  if (DotProduct = 0) AND (Distance > 0)
    then ray parallel to plane, no intersection
  if (DotProduct ≠ 0) AND (Distance = 0)
    then ray starts in plane, save intersection
  if (DotProduct ≠ 0) AND (Distance > 0)
    then ray intersects plane, determine intersection
    if closest so far save intersection
endloop
if (no intersections)
  then return OUTSIDE
calculate DotProduct of normal for ray best intersection and ray
calculate Distance from center of FaceA to FaceB
if (Distance = 0)
  then if (DotProduct > 0)
    then return SAME
    if (DotProduct < 0)
    then return OPPOSITE
if (DotProduct > 0)
  then return INSIDE
if (DotProduct < 0)
  then return OUTSIDE

```

Table 5.2: Labelling a face

#### 5.4.3.4 A Labelling Example

The intersection of two cubes example of Figure 5.2 requires four rays to be cast in order to label the entire object. The first two rays label the faces of ObjectA, and the second two the faces of ObjectB. The results of these four rays can be propagated to all





#### 5.4.3.6 A Classification Example

The intersection of two cubes example of Figure 5.2 will result in the polyhedral model shown in Figure 5.11. It is interesting to note that the boolean operation being performed does not affect the algorithm until the final stages. Therefore, once the splitting has been accomplished the table look-ups can be used to generate three different models (one for each operation). The results are shown in Figure 5.11

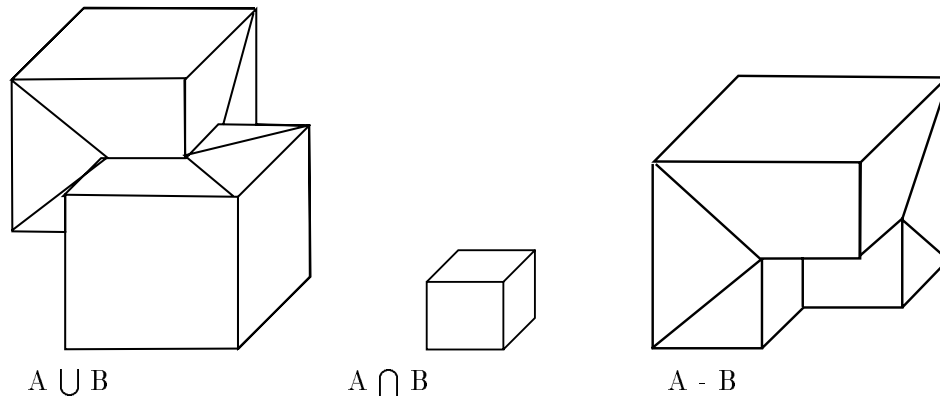


Figure 5.11: A Classification Example

#### 5.4.3.7 Merging Split Faces

The algorithm was chosen for many reasons relating to simplicity of implementation, how well it fit our needs, and how well its own requirements were met by the BRL/CAD models. There was a small problem with the algorithm that was not discovered until after the algorithm had been implemented.

The algorithm tends to over split faces of an object, and produce many faces for each resulting solid. Since we have no requirement that the resulting faces from the algorithm be convex, a merging phase was added to allow planar adjacent faces to be merged into one face. Once again, the B-rep stored the topological relationships needed for this phase of the algorithm.

The edge list of the B-rep is traversed. In a valid solid, every edge will be bounded by two faces. As the edge list is traversed, the two faces which share the edge are examined, and if their normals are equivalent (to within a tolerance), the faces are merged along the edge, and that edge is removed from the model. This phase greatly reduced the number

of faces in the resulting model. Figure 5.12 shows the resulting models for the two cube example.

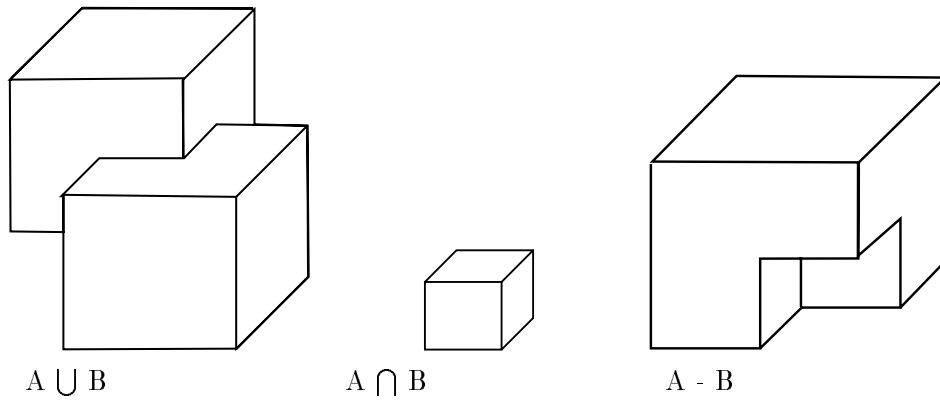


Figure 5.12: A Merge Example

#### 5.4.3.8 A Hybrid Approach

The fragmented triangle problem which results from over splitting a face of an object could be overcome by simple modifications to the original model. As the conversion algorithm was being tested, it became obvious that some floating point error problems could not be solved by the introduction of simple tolerance-mathematics (comparing the equality of two values to within a limit of certainty). It was discovered that slight modifications to the original model could be used to remove these problem areas. Thus a hybrid approach was taken in which the model was automatically converted, and problems were corrected with adjustment to the model.

#### 5.4.3.9 Results on a Vehicle Model

The algorithm has been applied to the M113 APC vehicle. Figure 5.13 shows four different views of the model shown in Figure 5.1b. Each face of the model is rendered in a unique color.



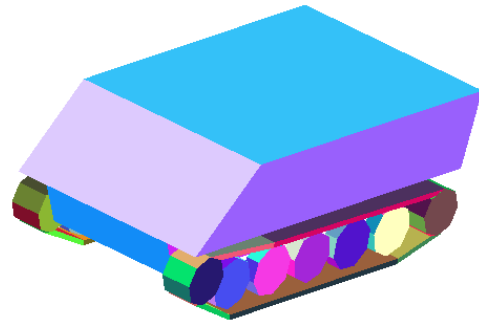
a. Top View



b. Side View



c. Rear View



b. 45° View

Figure 5.13: The M113 APC B-rep (See Color Plate 13)

## Chapter 6

### EXTRACTING 3D MODEL FEATURES

#### 6.1 Overview

After reducing the detail in the model to a level appropriate for recognition, the next step is to produce model features which are directly comparable with features extracted from sensor data. Our ATR algorithm dynamically updates the 3D position and orientation of the object model relative to the sensors, and observes the affect on an error measure. By searching for the lowest error possible, the optimal object match and pose can be discovered.

The ATR algorithm needs to be aware of two pieces of information for any given viewpoint: which silhouette edges are visible for matching to optical data, and the 3D synthetically sampled surface for matching to range data. In order to generate this information in real time, we will exploit the speed of modern graphics hardware. Since the ATR algorithm is initiated with an estimated viewpoint, it will make a request to the model feature extraction procedure. This process will then provide an initial set of visible features. If, during recognition, the viewpoint is modified significantly, the ATR algorithm can dynamically request the visible edges for the new viewpoint.

Earlier model feature extraction processes have focused on development of a model representation known as the aspect graph. Aspect graphs assume the model is centered at the origin of sphere which encapsulates the entire model. Every possible view of the model can then be represented as a point on that sphere. Aspect graphs are founded on the notion that there are regions on the view sphere in which viewpoints for a given region share a constant model topology [KvD76, KvD79]. The arcs in the graph represent movement from one set of visible features to another, and this characteristic has been

exploited in the ATR process [PD87], [Pla88]. Other techniques have used projection techniques similar to ours [Bon86, GM88, SJ89]. A more relevant discussion of feature extraction appears in [SD92], and describes a simple test we implement for determining whether or not an edge represents a silhouette edge. A complete discussion of related research can be found in Section 6.3

Our viewing geometry parameters are specified in spherical coordinates, with each viewpoint represented as an elevation ( $\phi$ ), a counterclockwise azimuth ( $\theta$ ), and a rotation degree ( $\gamma$ ). The ( $\phi$ ,  $\theta$ ) pair is used to form a vector towards the center of the model, which is centered at the origin of the sphere. The rotation ( $\gamma$ ) specifies an angle about that vector. Several other parameters are needed to specify the translation of the model in order to position the model in the different sensor coordinate systems. The ( $\phi$ ,  $\theta$ ,  $\gamma$ ) specifies the model orientation, and the translation ( $T_x, T_y, T_z$ ) moves the model to the correct position. These parameters are often used in graphics to specifying a 3D camera model and object viewing geometry.

## 6.2 Stored versus Demand Driven

Since the aspect graph is a commonly used method for representing visible model features, it is helpful to draw a comparison between our approach and that of the aspect graph. The aspect graph is a stored representation in which visible features are pre-computed for all relevant viewpoints. We have chosen instead to take a demand driven approach in which visible features are determined as they are needed. With modern graphics hardware able to render complex models efficiently, it makes sense to exploit this power. Since our algorithms rely heavily on this hardware, as its performance increases so does the performance of our algorithm.

However, our algorithm shares one important quality with the aspect graph: both need some procedure for generating the relevant features for a given viewpoint. Therefore, the same underlying algorithm can be used off-line to build up a stored aspect graph [KD87]. The more important issue is the nature and form of the algorithm used to generate the visible 3D features for different viewpoints. Several algorithms, not related to aspect graphs [AT81, Lee83], have been proposed to determine the visible portion of

a polygon from an arbitrary point. An algorithm for this task which meets our ATR requirements, and takes the demand driven approach, is presented below.

### 6.3 Related Research

The Aspect Graph was first introduced by J. J. Koenderink and A. J. van Doorn [KvD76], [KvD79]. Koenderink and van Doorn were concerned with developing a representation of shape that could be used directly for object recognition. Their graph representation stored visible surfaces for a given viewpoint node, and used arcs to represent adjacent viewpoints. The goal was to represent regions of constant aspect which could be matched against sensory inputs. However, their work was preliminary and left many issues unresolved.

In 1982, Indranil Chakravarty and Herbert Freeman presented a unique approach to model feature storage [CF82]. Their algorithm stored the projected model features of the 3D representation. The 2D projections were then used to match against 2D images. The format of the stored 2D projections was reminiscent of Koenderink and van Doorn's aspect graph representation in that Chakravarty and Freeman attempted to minimize the storage space necessary by storing the features hierarchically according to what they have termed *characteristic views*. Their representation was different but the motivation was the same. The main drawback to the Chakravarty and Freeman approach is that they store only 2D information, where as our algorithm requires 3D features.

In 1986, Luisa Bonfigliolo presented an algorithm that was geared more towards our needs [Bon86]. The algorithm generated the silhouette of any curved object which could be mapped into a polyhedral representation. The algorithm projected each line segment of the polyhedron onto an image plane and then intersected the projection with all the other projected segments to generate a continuous silhouette. We will be using a similar projection technique to obtain the visible silhouette segments. However, for our ATR application, they need not be continuous. In fact it will be the case that by forcing a continuous silhouette, several smaller line segments will be generated that may hinder the recognition process.

More recently at the University of Wisconsin, work has progressed on developing a feasible method for creating and manipulating an aspect graph. In [PD87], Harry Plantinga and Charles Dyer presented an algorithm for generating the aspect graph of polyhedral models. Their algorithm has been used in an object recognition system which utilizes a characteristic view approach. The approach is founded on the idea that if similar views of how an object will be perceived in the world are stored, the recognition process simply locates the correct stored view. The aspect graph they have developed is mainly concerned with the region boundaries at which a change in viewpoint causes a change in topology or occlusion of significant portions of formerly visible features. Areas in which the view point changes but topology does not are said to be regions of constant aspect.

Plantinga defines the view space partition (VSP) as a hyper-plane which divides viewing space into these regions of constant aspect. The aspect is considered to be the dual of the VSP representation, and they present an algorithm for switching between the two representations [Pla88]. Because the VSP is concerned with visible faces, an additional step is required to determine the visible silhouette edges. Refinements to the VSP algorithm would be needed if it were to be used for our ATR algorithm. However, we do not require a majority of the information the VSP method was developed to produce.

At the same time the VSP representation was being developed, Matthew R. Korn and Charles R. Dyer [KD87] derived a similar method for model feature extraction called a multi-view object representation. Their method was based on growing regions of consistent model topology on a view sphere. The algorithm begins with a uniform tessellation of a view sphere (an octahedron). At each iteration an individual face is subdivided into four new faces, which are stored in a tree as the children of the original cell. The splitting algorithm breaks down the view sphere until more subdivisions do not generate new model topologies, or a recursive limit has been reached. Region growing is then performed on the tree structure to merge nodes of similar topology. In order to speed up the region merging task, a unique indexing scheme allows the neighbors of any node in the tree to easily be determined. The merging technique results in a tree with the leaf nodes representing areas on the sphere with constant topology, and simple search techniques can be used to traverse the tree to discover all visible features for a given viewpoint.

Katsushi Ikeuchi presented a variation to the aspect graph called an interpretation tree [Ike87]. A string of bits is used to represent the faces visible from a particular view point, with each bit representing one face of the model. Initially a binary tree is constructed with each leaf node representing a direct view of a specific face (the face is fully visible from that position). Each branch in the tree represents a change in the set of visible features, and at each branch, the left branch has a bit string differing from the right branch by only one bit. The binary tree is developed as a part of their geometric modelling system, and is used to create an interpretation tree. Their interpretation tree consists of nodes representing constant model topology similar to an aspect graph. The storage requirements are much less than the other algorithms examined, and by applying simple binary logical operations to a string, it is immediately obvious which faces are visible for a viewpoint.

In 1988, Ziv Gigus and Jitendra Malik [GM88] present a slightly different algorithm for generating an aspect graph. Their method treated a polyhedron as a set of line vectors, and the faces formed from these lines are considered transparent. Based on simple vector computations, these lines can be projected onto a sphere, clipped against one another, and the resulting visible line segment can be determined. The algorithm they develop is reminiscent of Bonfigliolo's approach, and it too has the problem properly handling self occluding objects. The polyhedral objects we will use represent solid objects, and therefore do not work well with this simple approach.

The visible feature algorithms discussed so far all make the assumption that the underlying model can be represented with a set of polyhedra. In 1989, Thawach Sripradisvarakul and Ramesh Jain [SJ89], presented a different approach for generating aspect graphs that centered not around polyhedral objects, but instead dealt with continuous mathematical entities. Their presentation is pertinent because they develop the concept of *stable* and *non-stable* view points. A stable view point exists when slight perturbations in the viewing parameters do not cause dramatic changes in the visible model topology, where as perturbing a non-stable view point will cause a change. For the purposes of our models and our target recognition system it is not yet apparent how different the requests for visible features will be. It may become the case that the ATR algorithm finds these stable



areas, and may waste effort requesting redundant information (the model features will not change). However, we have chosen to deal with this problem in the implementation of the ATR algorithm, and neglect it in the feature extraction procedure.

W. Brent Seales and Charles Dyer [SD92] presented a refinement to the aspect graph representation that allowed for easy extraction of silhouette features. Their algorithm modifies Plantinga's VSP representation to form what they refer to as the rim appearance representation. This representation only stores the appearance of the silhouette, or the occluding contour, where as an aspect graph stores all possible model topologies as well as the transitions between these topologies. The rim method generates a subset of the information present in the aspect graph, and since the rim of a polyhedron is well-defined it can be computed for an arbitrary view point. They define the rim edge of a polyhedron as the line formed by two faces in which one is visible and the other is fully occluded. In addition, the edge must not be occluded by any other object. The rim appearance model is more in line with our requirements, only we will be using a different method to generate the silhouette. The rim method is also derived from the VSP method, and so utilization of the process would require a transition to the VSP model representation.

One of the main drawbacks of the aspect graph is its inability to deal with the problems of variable scale models. In sensor data, the scale of the object being recognized will be highly dependent upon the sensor viewing parameters, especially distance to the object. David W. Eggert has presented a version of the aspect graph to deal with the problem of different resolutions of the model in the sensor data [EBD<sup>+</sup>]. The algorithm we have developed deals with scale based on the viewing parameters of the sensors. In other words, the model is assumed to have the same dimensions as the object being recognized, and by using the same viewing parameters, scale will automatically be incorporated into the feature extraction process.

#### **6.4 The Model Extraction Algorithm**

Feature extraction techniques have typically centered around the notion of projecting each face of a model onto a viewing plane in order to determine which faces are visible for a specific view point. Our algorithm utilizes accelerated graphics hardware to increase the

efficiency of this operation. The main problem is not how to accomplish the projection, but rather how to interpret the projected results and obtain the 3D model features for use in ATR. Solving this reverse mapping problem turns out to be a non-trivial task.

The efficiency of our algorithm is dependent upon the capabilities of the accelerated graphics hardware it is running on. Currently, we are running the feature extraction algorithms on a Hewlett-Packard 715/50 CRX24Z, and a Sparc 10/24ZX. Both machines have 24 image planes. We initially chose to use the PEX graphics package as our interface to the accelerated hardware, but plan to move to the OpenGL package. The graphics package chosen does not affect the quality of the results, but does alter the speed at which they are obtained, the portability of the application to other platforms, and the ease with which the functionality can be implemented.

The algorithm breaks down into a three step process. The first step of the algorithm determines which faces are visible for a given viewpoint. The second phase uses this information to generate the model silhouette, and the third phase uses the visible faces to generate the sampled surface information.

#### 6.4.1 Determining Visible Faces

The first step in the model feature process is to label each face with a unique color. This color will provide a face id, and when the model is rendered the visible colors will be used to determine the corresponding face. A simple hashing algorithm is used to label each face with a color id: <sup>1</sup>

$$FaceColor = \lfloor \frac{MaxColor * FaceNum}{NumTotalFaces} \rfloor \quad (6.1)$$

For our application the MaxColor is 16,777,215 or  $2^{24} - 1$ , the maximum color for a 24 bit display. FaceNum is a unique number between zero and the total number of faces.

---

<sup>1</sup>A simpler mapping would be to assign the face color equal to the current face number. However the slight variation in color does not allow the human eye to easily distinguish the various faces. This method attempts spreads the colors across the entire range allow for more interpretable visual inspection of the model

Therefore, the only limit on the total number of faces allowed by our algorithm is the total number of colors available on the machine. Since most of our models are on the order of 500 faces, 24 bit color is more than adequate.

As each face is given its unique color, it is also stored in a color lookup table. Figure 6.1a shows the intersection of two cubes example used in Chapter 5, with the visible faces given unique colors. Figure 6.1b shows the corresponding B-rep <sup>2</sup> Figure 6.2 shows the color table used to index the faces based on their color <sup>3</sup>.

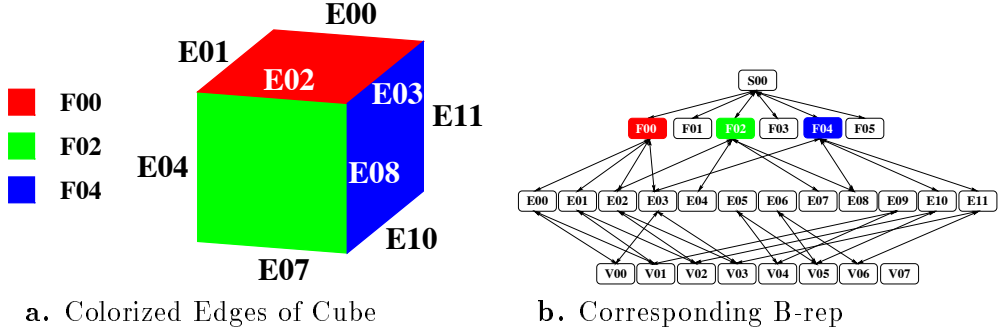


Figure 6.1: Assign a Unique Color to Each Face (See Color Plate 14)

After the color table has been created, each face is rendered, in its unique color, into an image plane with hidden surface removal. The viewing parameters used to render the model are based on the  $(\phi, \theta, \gamma)$  supplied by the ATR algorithm. This step is equivalent to the traditional feature extraction process of projecting each facet to determine the visible portion. We are using orthographic projection to render each face of the polyhedral model in its unique color. Orthographic projection is an adequate approximation since the vehicle being located will be a significant distance from the sensor in relation to the size of the model. Orthographic projection also simplifies the calculations used in the recovery of the 3D feature information.

The next step involves examining each pixel in the rendered image, and extracting the pixel color. If the color is not the same as a predetermined background color, the hash

---

<sup>2</sup>The back-faces of the cube for this viewpoint were not given a color in the figure so as to keep the figure simple.

<sup>3</sup>The edges for back-faces are also not listed

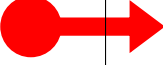
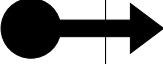
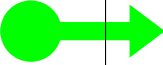

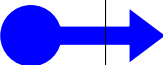

Color Table		Brep Data Structure	
FaceNum	Color		
0		Name: F00	Edges: E00, E01, E02, E03
1		Name: F01	
2		Name: F02	Edges: E02, E04, E07, E08
3		Name: F03	
4		Name: F04	Edges: E03, E08, E10, E11
5		Name: F05	

Figure 6.2: Color Table for Cube example (See Color Plate 15)

function is used to determine the appropriate index in the table for the face which caused the color. The reverse mapping function is:

$$FaceNum = \lfloor \frac{FaceColor * NumTotalFaces}{MaxColor} \rfloor \quad (6.2)$$

It was later determined that a slight round-off error can cause this calculation to fail. The problem is associated with most graphics programming languages, such as PEX, which only allow color specification as a floating point triple (RGB) in the range [0..1]. Therefore, when specifying the color to PEX for rendering, the color observed may be slightly off. Cursory examination has show the function fails once for every two hundred faces. In the case of failure, the cell to the left, and right are examined to find the correct face. Once the face is determined it is marked as *visible*. Another check is made to determine if the pixel in question is a silhouette pixel. A silhouette pixel is defined as any pixel in which any one of its eight neighbors are the background color. If the pixel was a silhouette pixel, the face is also marked as *silhouette*.

During the pixel examination process, a bitmap is created representing whether or not each pixel is a silhouette pixel. This bitmap will be used later to recover the 3D information lost when the image was rendered. Figure 6.3 shows the cube rendered in 3D and the corresponding bitmap.

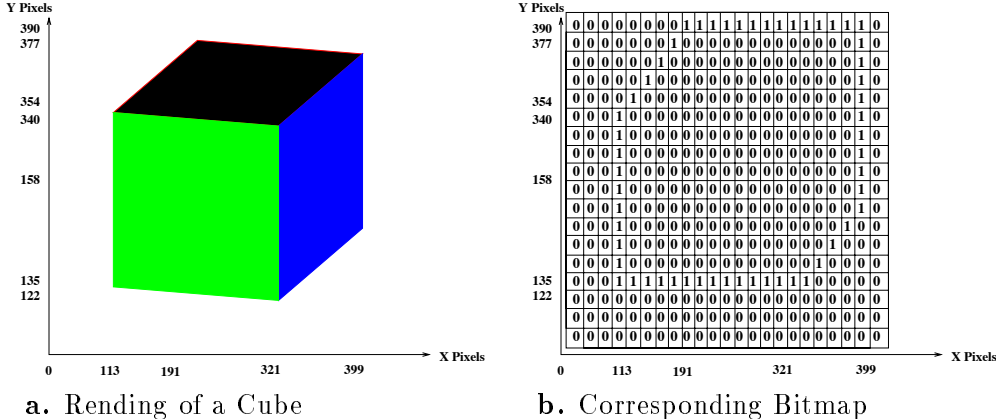


Figure 6.3: Rendering of Cube (See Color Plate 16)

### 6.4.2 Obtaining the Silhouette

After the faces which are visible and on the silhouette are determined, the next step is to obtain the 3D edges which caused the silhouette lines, and then clip them due to occlusion. Each edge in the model is examined. If only one of the faces to which the edge belongs is visible, and on the silhouette, that edge is marked as a silhouette edge. If both faces of an edge are visible then it is not possible for the edge to be on the silhouette [SD92]. Figure 6.4 shows the visible edges determined by this phase of the algorithm and the corresponding B-rep.

For the simple cube example, after the first phase the model silhouette has been found. However, for more complex models more work needs to be done. Due to occlusion, not all of a given edge may be visible for the given viewpoint. Therefore, portions of the line may need to be clipped due to occlusion. This is where the bitmap generated earlier becomes useful.

First all the vertices in the model are converted to pixel coordinates. A parametric representation is then used to march along the line formed by the two pixel endpoints. The line following algorithm begins at the pixel of the first vertex for the silhouette edge.

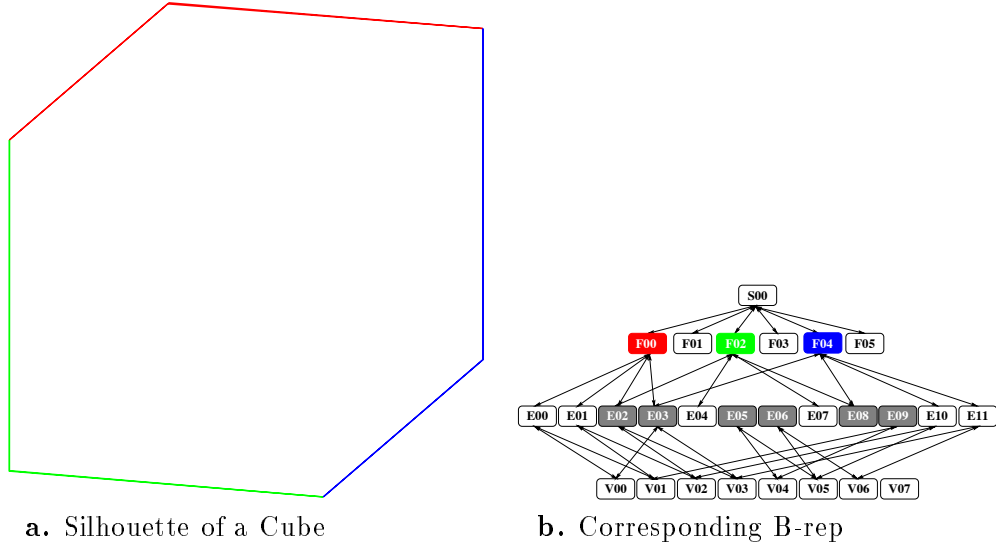


Figure 6.4: Silhouette of the Cube (See Color Plate 17)

At each distinct pixel along the line, a neighborhood around the pixel in the bitmap is examined. The traversal continues until an  $ON(1)$  bit is found in the bitmap. The color of the corresponding pixel in the image plane is examined and if it is the color of the face to which the edge belongs, it is used to compute a  $t$  value, called  $t_{in}$ , for the parametric line. This  $t$  value is stored as the beginning portion of the visible edge segment. The line following then continues until either the end point is reached, or an edge bit in the bitmap cannot be found. At this point, another  $t$  value is computed called  $t_{out}$ , and is stored as the exit point of the line. Due to the affine nature of orthographic projection, proportional distance is preserved between parameterized lines in 2D pixel and 3D model coordinates. We can therefore compute the parametric line in model coordinates, and use the  $t$  values obtain from the pixel parametric equation to obtain the approximate 3D endpoints of each visible edge segment. The lines are then given to the ATR algorithm as the visible silhouette features of the model for the given viewpoint.

The main problem to overcome in solving this reverse mapping problem from the 2D projected image to the 3D line segments was that of floating point error. Floating point imprecision can cause errors in the conversion from modelling to pixel coordinates. The rasterization results in jaggies and other aliasing problems that can lead the line following algorithm astray. To solve this problem, we were forced to increase the area

of the bitmap searched when clipping the visible edge. However, as the area searched increases, so does the number of edges incorrectly clipped. To address this problem, we added a user definable minimum Euclidean distance between the endpoints required for the line to be retained. For the images generated, we set this parameter to seven percent of the diagonal of the model bounding box. Another parameter was later added to remove a line when more than a certain percentage of the line is not visible.

The results after these modifications are shown in Figures 6.5b, 6.5c, and 6.5f. These three silhouette images are shown with the corresponding color image, and the viewing parameters were set to mimic those in the images. The red lines in the image represent the actual silhouette lines returned. The blue lines were first determined to be possible silhouette lines and were later removed for being either too short, or not actually on the silhouette. Since we are assuming we can extract perfect features from the model database, lines will be present in the silhouette image which are not present in the sensor image. For instance, in Figure 6.5c the bottom of vehicle is occluded by the terrain (grass). The model feature extraction is not aware of the terrain, and therefore produces lines not visible in the image.

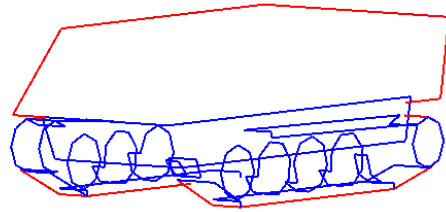
### **6.4.3 3D Sampled Surface**

The 3D silhouette is the appropriate representation for matching the model to optical data. However, the needs for range data are different. The features needed should be similar to the characteristics of the LADAR sensor used, thus reducing need for complex point to point matching. Similar features will be generated by sampling the surface of the model in a manner similar to the operation of an actual LADAR device.

In order to generate the the 3D sampled surface a ray casting technique is used. This technique mimics the geometry of the actual LADAR device used to obtain the sensor data. The algorithm first builds the LADAR geometry array. The geometry array is a set of ray direction vectors which mimic the directions an actual LADAR device would fire laser beams to determine depth values. The array is pre-computed and stored so as to reduce computation.



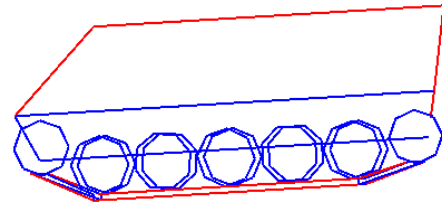
a. Full Image (720x480)



b. Corresponding Silhouette



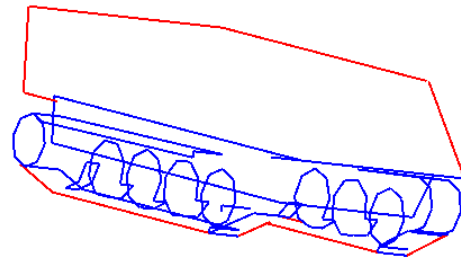
c. Full Image (720x480)



d. Corresponding Silhouette



e. Full Image (720x480)



f. Corresponding Silhouette

Figure 6.5: Silhouette for M113 APC (See Color Plate 18)



The next step involves transforming the model into the LADAR coordinate system. Since the ATR algorithm has already supplied the hypothesized matrix transformation ( $F_{m,r}$  from Chapter 2), this step involves just a simple matrix multiplication. Once the model is in the proper location, each ray in the LADAR geometry array is intersected with the model, and the closest intersection (if any) is recorded. This intersection provides similar information to what is actually determined by the LADAR device.

#### 6.4.3.1 LADAR Geometry

In order to generate the sampled surface quickly, an array of unit vectors in directions similar to the actual laser ranging device are created. The array is constructed by sweeping two parameters ( $\phi, \theta$ ) in the horizontal and vertical directions. The amount of horizontal rotation between each ray is  $0.0072^\circ$ , and the vertical rotation is  $0.1495^\circ$ . These two parameters are used to generate a vector with the following equations:

$$\phi = (0.5)(NumRows - i)(0.0072^\circ) \quad (6.3)$$

$$\theta = (0.5)(NumCols - j)(0.1495^\circ) \quad (6.4)$$

$$Ladar[i][j].x = \sin(\theta) \cos(\phi) \quad (6.5)$$

$$Ladar[i][j].y = \sin(\phi) \quad (6.6)$$

$$Ladar[i][j].z = \cos(\theta) \cos(\phi) \quad (6.7)$$

where  $i$  and  $j$  represent the position in the array being determined. This will generate a ray in the center of the array in roughly the  $\langle 0, 0, 1 \rangle$  direction. The other rays are rotated horizontally and vertically from that to generate the geometry shown in Figure 6.6a. The array dimension was set equivalent to the actual sensor used, which is 120 by 28.

#### 6.4.3.2 Generating Sampled Surfaces

After the visible faces have been obtained, the vertices are transformed into the LADAR sensor coordinate system. The  $(\phi, \theta, \gamma)$  supplied by the ATR algorithm are used for this process, as well as the translation vector to move the points into the LADAR

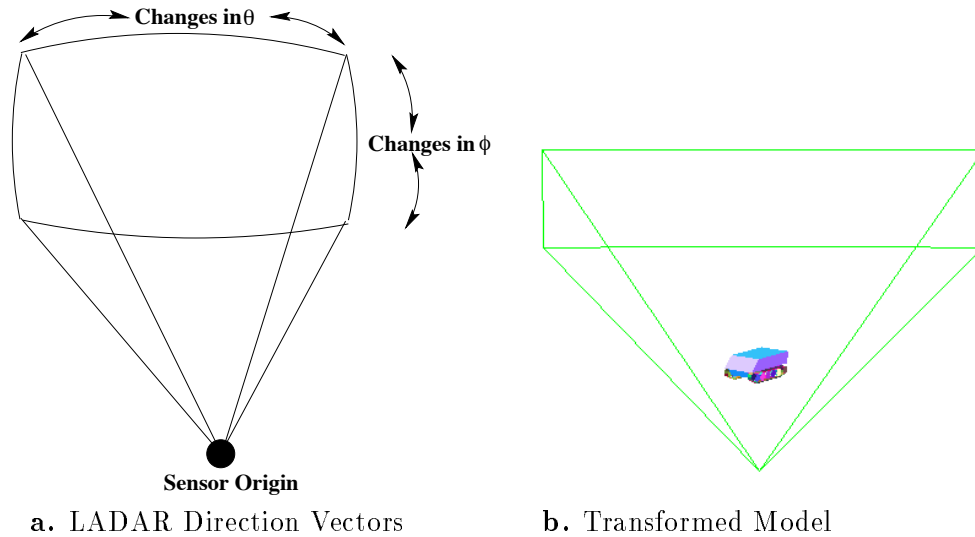


Figure 6.6: LADAR Geometry

coordinate system. This will effectively place the model into the sensor space as shown in Figure 6.6b. As the points are converted, the bounding box of the model (in LADAR coordinates) is determined, as well as the bounding box for each face (in LADAR coordinates).

The next phase involves intersecting all the rays in the LADAR vector array with all the faces in the model and recording the closest intersection. Several shortcuts have been taken to accelerate the process. First, the rays are intersected with the model bounding box. If no intersection occurs, the maximum depth of the sensor is returned. If the ray intersects the bounding box, then the ray is intersected with the bounding box of only those faces that have already been determined to be visible in the earlier phase of the algorithm. If the ray intersects both of these bounding boxes an intersection with the face is performed.

The intersection routine used is presented in [Hec94]. The process involves determining the ray intersection with the plane of the visible face. The point of intersection is then tested to make sure it lies within the bounds of the polygon. The computation involved in the intersection routines is greatly reduced due to all the rays beginning at the origin. This fact coupled with the steps to reduce the number of times the intersection needs to

be performed, allows the sampled surface information to be generated in an acceptable amount of time <sup>4</sup>.

### 6.4.3.3 Results

The results of the algorithm as applied to the data sets presented in Chapter 4 are shown in Figures 6.7a, 6.7c, and 6.7f. The actual data image is shown on the left, and the model sampled surface is shown on the right. The model samples are shown rendered with the colors related to their distance from the sensor (in the same color scheme as the actual LADAR data).

The representation used to generate the 3D sampled surface assumes a noiseless error model. The actual sensor obviously has some error in its sampled points. Our algorithm does not attempt to account for this noise, and therefore the results appear more uniform. Our algorithm also does not incorporate an atmospheric model which could account for the varied samples seen in the actual LADAR images. The error does not need to be accounted for in the the generation of the sensor data because it will be accounted for in the matching process. In the design of requirements for the extraction of the features from the model, it was determined the ATR algorithm would be responsible error reduction and noise modelling.

---

<sup>4</sup>On the order of 3 seconds for computing both the silhouette and sampled surface information



**a.** LADAR Image



**b.** Corresponding Sampled Surface



**c.** LADAR Image



**d.** Corresponding Sampled Surface



**e.** LADAR Image



**f.** Corresponding Sampled Surface

Figure 6.7: Sampled Surface for M113 APC (See Color Plate 19)

## Chapter 7

### FUTURE WORK

#### 7.1 Overview

This thesis has presented several different algorithms which can be applied to solid models. A conversion process is used to move the models from their native BRL/CAD format to an explicit polyhedral format. The polyhedra can then be used in a feature extraction process which provides information to an ATR algorithm. Each phase of the process was developed with one final goal in mind: to present the ATR algorithm with the necessary information for matching complex vehicle models to sensor data. There is room for expansion and improvement in each phase of the process, but the work presented here does meet the basic needs of our ATR work.

#### 7.2 Model Reduction

While a substantial amount of work on the conversion of the model from CSG to polyhedra has already been accomplished, there still remain refinements and modifications which need to be performed. The conversion algorithm is not yet completely reliable, in that modifications to the original model need to be performed before conversion process takes place. Several of the problems are fundamental to the process of converting a purely mathematical representation to the discrete representation of the computer. However, steps are being taken to improve the conversion process so as to increase its robustness and performance.

As of June 1995, several months after the initial conversion process was developed, BRL/CAD 4.4 was released. The new version of BRL/CAD supports a similar, but more robust implementation of the conversion process. The new algorithm is geared towards solving many of the problems associated with the implementation we choose (such as

decreasing the number of edges introduced and improving the numerical robustness). We are now using the new version to perform most of our model conversion.

The reduction phase of the algorithms still remains mostly a manual process. We are attempting to further automate the process to speed up the reduction of the models, and their subsequent conversion.

### **7.3 Model Analysis**

The current methods of extracting features from the 3D models were implemented as a first pass at providing the matching system with the information it needs to refine a pose estimate of an object in the sensor data. However, the matching system is not fully operational at this point. Therefore, the features chosen to be extracted, the model silhouette and sampled surface, were chosen based on what was believed to be the most stable approach to solving the problem. As the matching system comes on-line, a more detailed analysis of the matching system performance based on the current features will be needed.

### **7.4 Sensor Data Feature Extraction**

A system for extracting features from the model database for a hypothesized viewpoint has already been developed. The next step is to extract corresponding features from the sensor data. A system to perform just this task is already being developed. The approach is unique in that it is using the features already extracted from the model to drive the location of corresponding features in the sensor data.

### **7.5 Integration with the ATR Algorithm**

The last and final step is to integrate all of the pieces of the system together, and run the complete ATR algorithm on various test sensor suites and with various models. This process is also currently underway and expected to be near completion by the end of the year (1995).

## Chapter 8

### CONCLUSION

The work presented in this thesis was driven by the needs of the ATR algorithm being developed simultaneously at Colorado State University. The project began as a spin-off of the Co-Registration work when it became apparent that there was no direct method for extracting information from the model. Thus began the pursuit of a viable method for extracting information from the model for a hypothesized model position.

When the process started, it was immediately obvious that the BRL/CAD models were highly detailed and in a format not conducive to the type of extraction we were going to perform. A method to reduce these models, as well as convert them to a more useful format for our application, was then developed. In order to expedite this process, a large amount of user interaction was required, and there is still an on-going effort to automate this process. The end result was a method to produce a model which could then be used by a feature extraction algorithm.

Once the model was in a format where the information needed by the algorithm was stored explicitly, we needed to determine how to produce features comparable to those present in the sensor data. The algorithm needed to produce features to compare against both optical and range imagery. We made the assumption that the most stable optical features would be the silhouette, and the most stable range features the sampled surface. Algorithms to extract both pieces of information were then presented.

The results of the entire process have allowed the creation of a system which produces a model in format which allows the retrieval of features directly comparable to the information provided by the sensor data. Now that models in a form required for ATR are available, work begins testing these ATR algorithms. The testing process will most surely lead to a better understanding of the best features needed for model matching, and thus to refinements in the model features extraction process.

## REFERENCES

- [AN90] J. K. Aggarwal and N. Nandhakumar. Multi-sensor fusion for automatic scene interpretation. In Ramesh C. Jain and Anil K. Jain, editors, *Analysis and Interpretation of Range Images*. Springer-Verlag, 1990.
- [AT81] David Avis and Godfried T. Toussaint. An optimal algorithm for determining the visibility of a polygon from an edge. *IEEE Transactions on Computers*, c-30(12):910–914, December 1981.
- [BDHR94] Shashi Buluswar, Bruce A. Draper, Allen Hanson, and Edward Riseman. Non-parametric Classification of Pixels Under Varying Outdoor Illumination. In *Proceedings: Image Understanding Workshop*, pages 1619–1626, Los Altos, CA, November 1994. ARPA, Morgan Kaufmann.
- [Bes88] Paul J. Besl. Geometric modeling and computer vision. *Proceedings of the IEEE*, 76(8):936–958, August 1988.
- [Bev93] J. Ross Beveridge. *Local Search Algorithms for Geometric Object Recognition: Optimal Correspondence and Pose*. PhD thesis, University of Massachusetts at Amherst, may 1993.
- [BHP94] J. Ross Beveridge, Allen Hanson, and Durga Panda. Integrated color ccd, flir & ladar based object modeling and recognition. Technical report, Colorado State University and Alliant Techsystems and University of Massachusetts, April 1994.
- [BHP95] J. Ross Beveridge, Allen Hanson, and Durga Panda. Model based sensor fusion of flir, color and ladar. In *SPIE: Sensor Fusion and Networked Robotics VIII*, Philadelphia, October 1995. SPIE. to appear.
- [BJLP92] James Bevington, Randy Johnston, Joel Lee, and Richard Peters. A modular target recognition algorithm for ladar. In *Proceedings of the 2nd Automatic Target Recognizer Systems and Technology Conference*, pages 91 – 104, Fort Belvoir, VA, mar 1992.
- [Bon86] Luisa Bonfigliolo. An algorithm for silhouette of curved surfaces based on graphical relations. *Computer-Aided Design*, 18(2):95–101, March 1986.
- [BPY94] J. Ross Beveridge, Durga P. Panda, and Theodore Yachik. November 1993 Fort Carson RSTA data collection final report. Technical Report CS-94-118, Computer Science Dept., Colorado State University, January 1994.
- [CF82] Indranil Chakravarty and Herbert Freeman. Characteristic views as a basis for three-dimensional object recognition. *SPIE: Robot Vision*, 336:37–45, 1982.



- [EBD<sup>+</sup>] David W. Eggert, Kevin W. Bowyer, Charles R. Dyer, Henrik I. Christensen, and Dmitry B Goldgof. The scale space aspect graph. NSF grant IRI-8817776.
- [FH87] Pascal Fua and Andrew J. Hanson. Using generic geometric models for intelligent shape extraction. In *American Association for Artificial Intelligence Conference*, pages 706–711, 1987.
- [Fre95] Kellye C. Frew. Modeling target damage in effectiveness / vulnerability assessments in three dimensions (eva-3d). In Keith Applin, editor, *BRL-CAD Symposium 95*, volume APG-EA. Army Research Laboratory, June 1995.
- [FvDFH90] James D. Foley, Andries van Dam, Steven K. Fiener, and John F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley Publishing Company, 2 edition, 1990.
- [GBSF94] Michael E. Goss, J. Ross Beveridge, Mark Stevens, and Aaron Fuegi. Visualization and verification of automatic target recognition results using combined range and optical imagery. In *Proceedings: Image Understanding Workshop*, pages 491–494. ARPA, nov 1994.
- [GBSF95] Michael E. Goss, J. Ross Beveridge, Mark R. Stevens, and Aaron D. Fuegi. Three-dimensional visualization environment for multi-sensor data analysis, interpretation, and model-based object recognition. In Georges G. Grinstein and Robert F. Erbacher, editors, *Proceedings: Visual Data Exploration and Analysis II*, pages 283–291. SPIE Vol. 2410, feb 1995.
- [GJSL90] Jeffrey Gillberg, Randy Johnston, Kris Siejko, and Joel Lee. Laser radar atr algorithms. Technical Report DAABO7-87-C-F109, Honeywell Systems and Research Center, Minnesota, May 1990.
- [GM88] Ziv Gigus and Jitendra Malik. Computing the aspect graph for line drawings of polyhedral objects. *IEEE Proceedings on Computer Vision and Pattern Recognition*, pages 654–661, 1988.
- [Hec94] Paul S. Heckbert. *Graphics Gems IV: A Collection of Practical Techniques for the Computer Graphics Programmer*. Academic Press, 1994.
- [Ike87] Katsushi Ikeuchi. Precompiling a geometrical model into an interpretation tree for object recognition in bin-picking tasks. In *Proc. DARPA Image Understanding Workshop*, pages 321–330, February 1987.
- [KD87] Matthew R. Korn and Charles R. Dyer. 3d multi-view object representations for model-based object recognition. *Pattern Recognition*, 20(1):91–103, 1987.
- [KvD76] J.J. Koenderink and A.J van Doorn. The singularities of visual mapping. *Biological Cybernetics*, 24:51–59, 1976.
- [KvD79] J.J. Koenderink and A.J van Doorn. The internal representation of shape with respect to vision. *Biological Cybernetics*, 32:211–216, 1979.
- [Lee83] D. T. Lee. Visibility of a simple polygon. *Computer Vision, Graphics, and Image Processing*, 22:207–221, 1983.

- [LTH86] D.H Laidlaw, W.B. Trumbore, and J.F. Hughes. Constructive solid geometry for polyhedral objects. *Computer Graphics*, 20(4):161–170, August 1986.
- [Man90] Martti Mantyla. *An Introduction to Solid Modeling*. Computer Science Press, 1990.
- [NAT90] Bruce Naylor, John Amanatides, and William Thibault. Merging bsp trees yields polyhedral set operations. *Computer Graphics*, 24(4):115–124, August 1990.
- [PD87] Harry Platinga and Charles Dyer. Visibility, occlusion, and the aspect graph. Technical Report 736, University of Wisconsin - Madison, December 1987.
- [Pla88] William Harry Plantinga. *The ASP: A Continuous, Viewer-Centered Object Representation for Computer Vision*. PhD thesis, University of Wisconsin at Madison, 1988.
- [Pop94] Arthur R. Pope. Model-based object recognition. Technical report, University of British Columbia, January 1994.
- [PS86] L. K. Putnam and P. A. Subrahmanyam. Boolean operations on n-dimensional objects. *IEEE Computer Graphics and Applications*, 6(6):43–51, June 1986.
- [RTKM89] Steven K. Rodgers, Carl W. Tong, Matthew Kabrisky, and James P. Mills. Multi-sensor fusion of ladar and passive infrared imagery for target segmentation. *Optical Engineering*, 28(8):881–886, August 1989.
- [RV80] A.A.G. Requicha and H.B Voelcker. Constructive solid geometry. Technical Report TM-25, University of Rochester, April 1980. Production Automation Project Technical Memorandum.
- [RV83] A.A.G. Requicha and H.B Voelcker. Solid modeling: Current status and research directions. *IEEE Computer Graphics and Applications*, 3(7), October 1983.
- [RV85] A.A.G. Requicha and H.B Voelcker. Boolean operations in solid modeling: Boundary evaluation and merging algorithms. *Proceedings of the IEEE*, 73(1), January 1985.
- [SB94] Anthony N. A. Schwickerath and J. Ross Beveridge. Object to multi-sensor co-registration with eight degrees of freedom. In *Proceedings: Image Understanding Workshop*, pages 481–490. ARPA, nov 1994.
- [SD92] W. Brent Seales and Charles R. Dyer. Modeling the rim appearance. In *Proceedings of the 3rd International Conference on Computer Vision*, pages 698–701, 1992.
- [SJ89] Thawach Sripradisvarakul and Ramesh Jain. Generating aspect graphs for curved objects. In *Proceedings of the IEEE Workshop on Interpretation of 3D Scenes*, pages 109–115. IEEE, 1989.

- [Sny92] John M. Snyder. *Generative Modeling for Computer Graphics and CAD: Symbolic Shape Design Using Interval Analysis*. Academic Press, first edition, 1992.
- [SWF95] G.D Sullivan, A.D. Worrall, and J.M Ferryman. Visual object recognition using deformable models of vehicles. In *Workshop on Context-Based Vision*, pages 75–86, june 1995.
- [U. 91] U. S. Army Ballistic Research Laboratory. *BRL-CAD User's Manual*, release 4.0 edition, December 1991.
- [Vat92] Bala R. Vatti. A generic solution to polygon clipping. *Communications of the ACM*, 35(7):57–63, July 1992.
- [VDL94] Jacques G. Verly, Dan E. Dudgeon, and Richard T. Lacoss. Model-based automatic target recognition system for the UGV/RSTA ladar. In *Proceedings: Image Understanding Workshop*, pages 559–583. ARPA, November 1994.
- [WA77] K. Weiler and P. Atherton. Hidden surface removal using polygon area sorting. *Computer Graphics*, 11(2):214–222, Summer 1977.

## Appendix A

### ACRONYMS

Acronym	Meaning
2D	Two Dimensional
3D	Three Dimensional
APC	Armored Personnel Carrier
ARPA	Advanced Research Projects Agency
B-Rep	Boundary Representation
BRL/CAD	Ballistic Research Laboratory Computer Aided Design
BSP	Binary Space Partition
CSG	Constructive Solid Geometry
CSU	Colorado State University
FLIR	Forward Looking Infrared Red
FOV	Field-Of-View
LADAR	Laser and Depth Ranging
MGED	Multi-display Graphical EDitor
R-Set	Regularized-Set
RGB	Red-Green-Blue
RSTA	Reconnaissance, Surveillance and Target Acquisition
UGV	Unmanned Ground Vehicle
VSP	View Space Partition

**Appendix B**

**COLOR PLATES**