



National Library  
of Canada

Bibliothèque nationale  
du Canada

Acquisitions and  
Bibliographic Services Branch

Direction des acquisitions et  
des services bibliographiques

395 Wellington Street  
Ottawa, Ontario  
K1A 0N4

395, rue Wellington  
Ottawa (Ontario)  
K1A 0N4

Your file    *Voire référence*

Our file    *Notre référence*

## NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

## AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

# Extracting Polyhedral Models From A Range Image — A Hybrid Approach

by

Xiao Ming Zhu

A THESIS SUBMITTED IN PARTIAL FULFILLMENT  
OF THE REQUIREMENTS FOR THE DEGREE OF  
MASTER OF APPLIED SCIENCE

in the School  
of  
Engineering Science

© Xiao Ming Zhu 1991

Simon Fraser University

October, 1991

*All rights reserved. This work may not be reproduced  
in whole or in part, by photocopy or  
other means, without permission of the author.*



National Library  
of Canada

Bibliothèque nationale  
du Canada

Acquisitions and  
Bibliographic Services Branch

Direction des acquisitions et  
des services bibliographiques

395 Wellington Street  
Ottawa, Ontario  
K1A 0N4

395, rue Wellington  
Ottawa (Ontario)  
K1A 0N4

*Your file* *Votre référence*

*Our file* *Notre référence*

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-78265-X

Canada

# APPROVAL

NAME: Xiao Ming Zhu  
DEGREE: Master of Applied Science  
TITLE OF THESIS: **Extracting Polyhedral Models From  
A Range Image: A Hybrid Approach**

## EXAMINING COMMITTEE:

Chairman: Dr. John Jones

---

~~\_\_\_\_\_~~  
Dr. Kamal Gupta, Senior Supervisor

---

~~\_\_\_\_\_~~  
Dr. Tom Calvert, Supervisor

---

~~\_\_\_\_\_~~  
Dr. Vladimir Cuperman, Supervisor

---

~~\_\_\_\_\_~~  
Dr. Ze-Njan Li, Supervisor

---

~~\_\_\_\_\_~~  
Dr. John Dill, Examiner

DATE APPROVED:

---

26 Nov., 1991

---

PARTIAL COPYRIGHT LICENSE

I hereby grant to Simon Fraser University the right to lend my thesis, project or extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Thesis/Project/Extended Essay

"Extracting Polyhedral Models from Range Images - a Combined Approach"

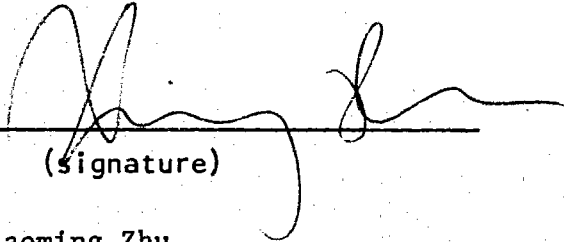
---

---

---

---

Author:

  
\_\_\_\_\_  
(signature)

Xiaoming Zhu

\_\_\_\_\_  
(name)

Nov. 20, 1991

\_\_\_\_\_  
(date)

# ABSTRACT

In many robotic tasks such as collision avoidance and grasping, 3-dimensional models of polyhedral objects are needed. These models are in the form of 3-dimensional faces, edges, vertices and their geometry and topology. This thesis provides a new hybrid approach to extract such a 3-dimensional model of the visible portion of a polyhedral object in a range image. It presents a unique approach called “combine and compare” to obtain robust analytical descriptions of faces, edges and vertices. First, straightforward edge-based methods are used to extract jump and roof edge maps. Then, region-based methods in conjunction with least-squares-fit techniques are used to extract analytical descriptions of the planes that form the faces of the object. All the plane descriptions are then intersected (“combined”) to obtain possible roof edge segments. These segments are then validated (“compared”) with the roof edge map to derive the actual edge segments. A similar “combine and compare” process is used to derive the vertices from the edge segments. These faces, edges and vertices are combined to yield a solid model of the polyhedral object. Using region-based information also facilitates a simpler processing of the vertices than existing pure edge-junction based approaches. We view this work as a step toward the larger goal of obtaining complete 3-dimensional boundary representations of objects in a scene from multiple range images.

*To My Parents*

# ACKNOWLEDGEMENTS

I would like to express my appreciation to my senior supervisor Dr. Kamal Gupta for his assistance during this research.

I am very grateful to Frank Tong, Zhenping Guo, Louis Brassard, Gilles Dionne, William Li and many others for their valuable discussions on my thesis. Range images provided by the PRIP Lab at Michigan State University are also gratefully acknowledged and in particular thanks are owed to Dr. Patrick Flynn.

Finally, special thanks go to my family and friends for their encouragement and support throughout the course of this thesis.



# CONTENTS

APPROVAL . . . . .	ii
ABSTRACT . . . . .	iii
ACKNOWLEDGEMENTS . . . . .	v
LIST OF FIGURES . . . . .	xi
LIST OF TABLES . . . . .	xii
ABBREVIATIONS . . . . .	xiii
1 INTRODUCTION . . . . .	1
1.1 Motivation . . . . .	1
1.2 Overview of Our Research Work . . . . .	5
1.3 Why assume a polyhedral world? . . . . .	9
1.4 Why a Combined Approach? . . . . .	9
1.4.1 Pro and Cons of Edge Based Method . . . . .	9
1.4.2 Pro and Cons of Region Based Methods . . . . .	10
2 Background And Literature Review . . . . .	13
2.1 Range Image Processing . . . . .	14
2.1.1 Edge Detection . . . . .	14
2.1.1.1 Jump Edge . . . . .	14

2.1.1.2	Roof Edge . . . . .	14
2.1.2	Hough Transform . . . . .	15
2.1.3	Segmentation . . . . .	16
2.2	Range Image Geometric Analysis . . . . .	19
3	The Description and Implementation of the Algorithm . . . . .	25
3.1	Method Overview . . . . .	25
3.2	The Algorithm . . . . .	26
3.2.1	Jump Edge Map . . . . .	26
3.2.2	Hough Transform on Jump Edges . . . . .	31
3.2.3	Surface Normal Calculation . . . . .	34
3.2.4	Roof Edge Map . . . . .	36
3.2.5	Surface Segmentation and Description . . . . .	37
3.2.5.1	Surface Segmentation . . . . .	39
3.2.5.2	Region Shrinking . . . . .	42
3.2.5.3	Generating Surface Equations . . . . .	42
3.2.6	Jump Edge Labeling . . . . .	44
3.2.7	Roof Edge Segmentation and Labeling . . . . .	45
3.2.8	Extracting Vertices . . . . .	51
3.2.9	Closed Loop Description . . . . .	57
3.3	Summary . . . . .	60
4	Experimental Results and System Evaluation . . . . .	61
4.1	A Simple Polyhedral Object . . . . .	62
4.2	A Polyhedral object with Vertical and Horizontal Lines . . . . .	66
4.3	A Polyhedral Object with a Non-Convex Surface . . . . .	71

5	Conclusion . . . . .	82
	REFERENCES . . . . .	88

# LIST OF FIGURES

1.1	A Robot Workcell . . . . .	3
1.2	The input to the proposed system: a range image . . . . .	6
2.1	Herman's approach . . . . .	22
3.1	An Overview of Our Approach . . . . .	27
3.2	Overview of the proposed algorithm . . . . .	28
3.3	Overview of the implementation procedures . . . . .	29
3.4	Edge Operators Used . . . . .	30
3.5	Gradient Operator Used . . . . .	34
3.6	Roof Edge Operator Used . . . . .	36
3.7	Remove false roof edges . . . . .	38
3.8	Selecting the reference surface normal . . . . .	40
3.9	Illustration: shrink the region . . . . .	43
3.10	Illustration . . . . .	45
3.11	Illustration . . . . .	46
3.12	Illustration . . . . .	50
3.13	Illustration: a real vertex that are obvious . . . . .	52
3.14	Illustration: a false vertex that is outside a region . . . . .	52

3.15	Illustration: a false vertex that is inside the region . . . . .	53
3.16	Illustration: a false vertex that is on the edge . . . . .	53
3.17	Illustration: an example of a valid vertex II. . . . .	54
3.18	Illustration . . . . .	58
3.19	Illustration . . . . .	59
4.1	“BOX” Image: Original Z Image . . . . .	62
4.2	“BOX” Image: Flag Image . . . . .	62
4.3	“BOX” Image: Jump Edge Map . . . . .	63
4.4	The Process of Getting the Roof Edge Map . . . . .	64
4.5	“BOX” Image: Combined Edge Map . . . . .	66
4.6	“Box” Image: Region Clustering . . . . .	67
4.7	“BOX” Image: Region Shrinking . . . . .	68
4.8	Result: Box Image . . . . .	69
4.9	“Rectangular” Image: Original Z Image . . . . .	69
4.10	“Rectangular” Image: Flag Image . . . . .	70
4.11	“Rectangular” Image: Jump Edge Map . . . . .	70
4.12	The Process of Getting the Roof Edge Map . . . . .	72
4.13	“Rectangular” Image: Combined Edge Map . . . . .	73
4.14	“Rectangular” Image: Region Segmentation . . . . .	74
4.15	“Rectangular” Image :Region Shrinking . . . . .	75
4.16	“L” Image: Original Z Image . . . . .	76
4.17	“L” Image: Flag Image . . . . .	76
4.18	“L” Image: Jump Edge Map . . . . .	76

4.19	“L” Image: The Process of Getting the Roof Edge Map . . . . .	77
4.20	“L” Image: Combined Edge Map . . . . .	78
4.21	“L” Image: Region Segmentation . . . . .	79
4.22	“L” Image: Region Shrinking . . . . .	80
4.23	“L” Image: an interesting example . . . . .	81
5.1	Example: One surface has no more than one missing edge . . . . .	86
5.2	Example: One surface has more than one missing edge . . . . .	87

# LIST OF TABLES

1.1	The output of the proposed system: a boundary representation. . . .	7
3.1	The structure used in the HT to get the edge information . . . . .	31
3.2	The pseudo-code for the algorithm for tracing in the HT . . . . .	32
3.3	The pseudo-code for the HT used in this thesis . . . . .	33
3.4	The pseudo-code for surface segmentation . . . . .	41
3.5	The pseudo-code for surface segmentation . . . . .	48
3.6	The structure used to get the roof edge information. . . . .	49
3.7	The pseudo code for vertex confirmation . . . . .	57
4.1	The result of the processing . . . . .	65

# ABBREVIATIONS

B-Rep	Boundary Representation
HT	Hough Transform
LSF	Least Squares Fit



# CHAPTER 1

## INTRODUCTION

### 1.1 Motivation

A main goal of robotics research is to make future generations of robots considerably more autonomous than present robotic systems. To achieve this goal, these robots must be able to manipulate and interact with the environment. For instance, they must be able to automatically grasp an object; automatically plan their path to avoid obstacles in their way, etc. To perform such tasks, a robot needs a 3-dimensional geometric model of its environment in terms of faces, edges and vertices and their geometry and topology, e.g., which vertices form a face.

Currently, it is assumed that the data necessary for constructing such a model can be derived from CAD models which are generated interactively. However, for autonomous operation, it is important to automatically acquire these geometric models for various tasks such as path planning, grasping, etc. In fact, such robotic

application has opened a new area of computer vision research which has long been focusing only on recognition problems. It emphasizes more on the description of the scene and various image processing techniques to achieve this. Recognition is irrelevant to many of these robotic applications.

Although a great deal of research in machine vision has been concerned with acquiring 3-dimensional depth information (shape from X) [21], from intensity images, recently, direct depth measurements (hereafter called a range image) of a scene are available with a device called the laser range-finder [Rioux]. These devices provide the depth data explicitly and thereby avoid many complex problems inherent in extracting this depth information from intensity images. For our purposes, a range image is a large matrix,  $z(i, j)$  of distances from a sensor coordinate system to the surface points on objects in a scene[3].)

Because range images explicitly provide depth, some may wonder why do we still need range image processing. Don't we already know where the object is? Yes, we know it at the pixel level. However, we still need to extract meaningful descriptions from it. For instance, we may be interested in finding the shortest path for a manipulator arm around a set of obstacles. We need to know more than pixel level information to plan a path that achieves this goal. Similarly, to find a grasp position on a polyhedral object, one needs to reason about the relationship among the faces of the object. Therefore, a more symbolic 3-dimensional geometric description of the scene from range image is required.

This thesis is a step toward automatically extracting 3-dimensional geometric models from range images. Imagine a robot work cell as shown in Figure 1.1 for an example. It consists of a robot arm, a laser range finder and many polyhedral

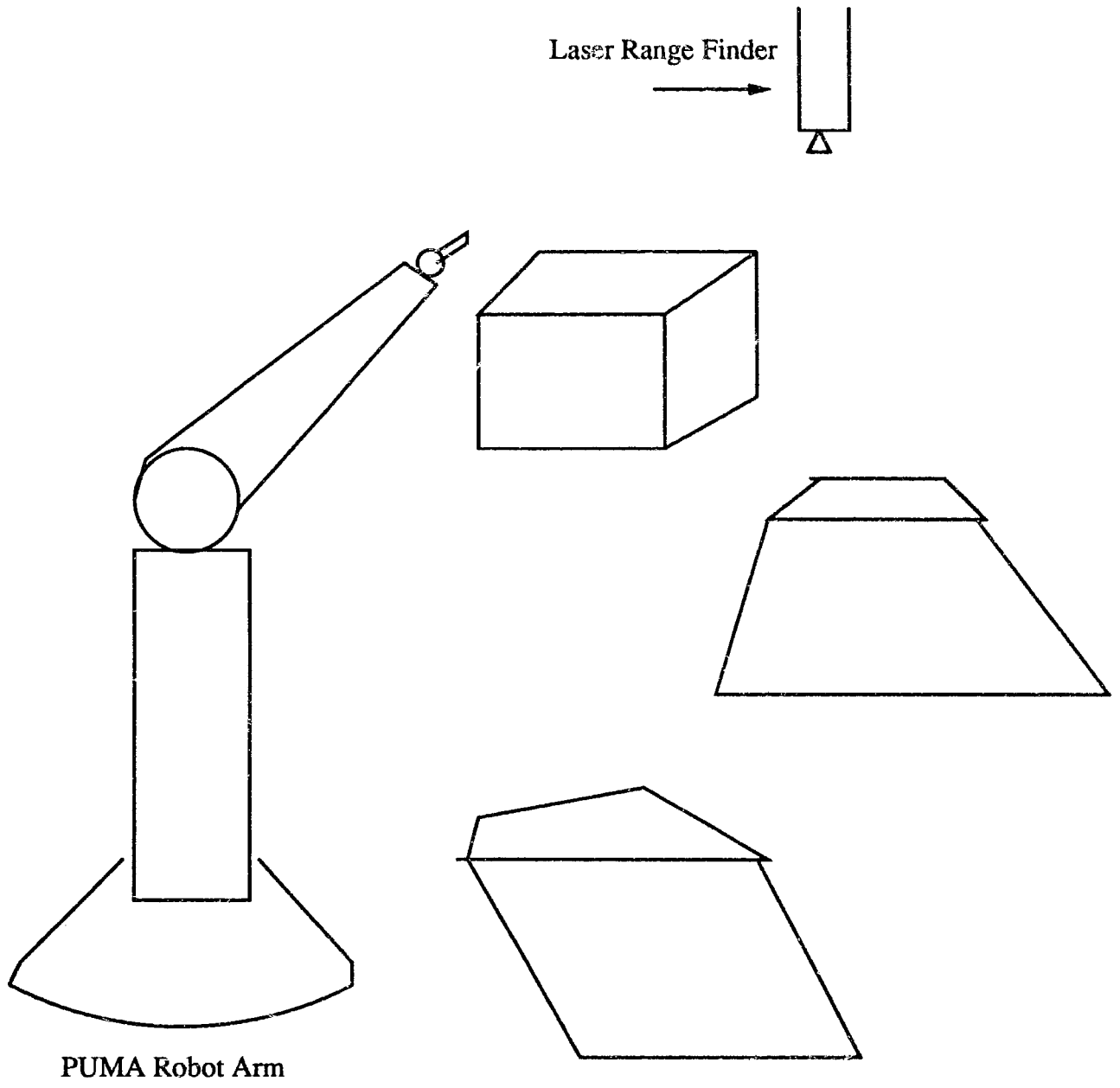


Figure 1.1: A Robot Workcell

objects. In robotics literature, many algorithms have been developed for planning collision-free paths for the robot arm, given a geometric description of the objects in its environment [13], [14] and [29]. Our ultimate aim is to extract such geometric descriptions from a series of range images taken from different viewpoints by the scanner.

Clearly, it is a tall order. As a first step, we would like to extract geometric descriptions of visible portions of an object from a single range image. This thesis provides one way to do it.

Having stated the general goal of the thesis, let me elaborate on what is meant by a 3-dimensional geometric model for path planning. A basic paradigm in path planning, called the configuration space approach, explicitly computes the constraints on the joint angles of the robot arm. For such computations, it needs the following information (i) which edges form a face, and the outward normal to the face, (ii) which faces are adjacent, (iii) coordinates of the vertices. Such representations are very close to what are called boundary representations (B-Rep) in Solid Modeling literature. A boundary representation is a method of representing the 3-D geometry of the edges and nodes of an object without a full surface representation.(refer to [6].) We will use the terms polyhedral models, or geometric models, or boundary representations to mean the same thing. In this thesis, the boundary representation is stored as vertices in a counter-clockwise order (with the inside of the face on the left hand side as the edges are traversed) on each face. Here vertex coordinates provide the geometric data while the counter-clockwise order provides the outward normal of the face. This representation gives the adjacency relationships of all vertices and edges. Although our methods are do recover the the surface adjacency, it

is not made explicit in this representation.

Up until now, most vision research has focused on recognition problems. Such complete geometric descriptions are not necessarily required for recognition; they are, however, required for interaction with the environment, e.g., grasping and obstacle avoidance.

Our aim, in this thesis, is to extract boundary representations of visible surfaces of a polyhedral object from its range image. The ultimate aim, of course, is to provide a range-image based vision system for a robotic work cell to build a 3-dimensional boundary description of the objects in the robot's environment. This description will then be used by robotic path planning algorithms to find a collision-free path for the robot.

## 1.2 Overview of Our Research Work

By its very nature, this research requires the techniques and knowledge from low level image analysis (edge detection, segmentation, etc.) and uses geometric computations (we call it geometric analysis) to extract the boundary representation of an object in the scene. It combines edge-based and region-based methods to get such a description. The edge-based approach is used to get the jump edge map<sup>1</sup>, the roof edge map<sup>2</sup> and symbolic jump edge segments. The region-based approach is used

---

<sup>1</sup>Discontinuity in depth

<sup>2</sup>Discontinuity in surface orientation

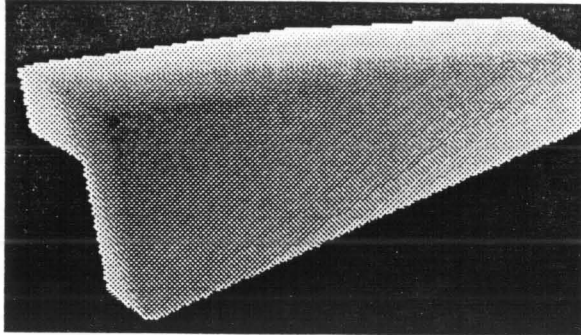
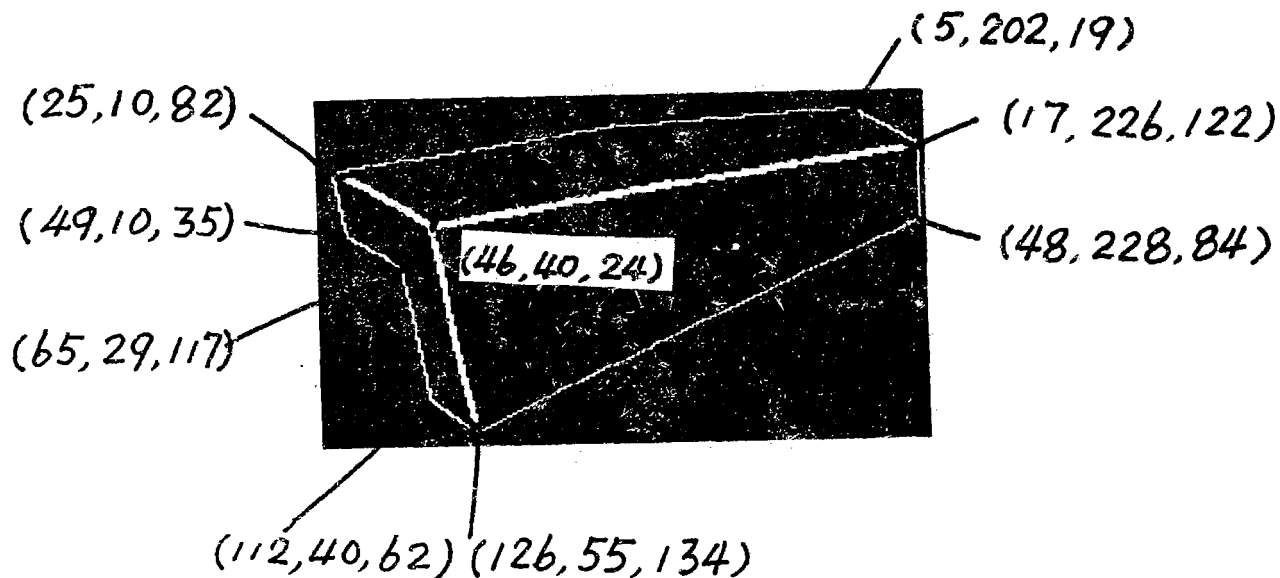


Figure 1.2: The input to the proposed system: a range image

to get partitioned surfaces, surface equations and other information about surfaces. While most other researchers have mainly focussed on improving the edge-based methods, such as the Hough transform (HT) [11], to get a more reliable surface information, this thesis uses the region-based method to get the surface information directly from the raw data. It then combines all the surface equations to get all possible roof edges. Later, it compares the possible edges detected above with the original roof edge map to remove the false roof edges. This unique way to combine the above two approaches is not only used to get the roof segmentation but also vertex location. Here is an example of what the input and output of the system look like. Figure 1.2 shows the input to the system, a range image, and Table 1.1 gives the output, a boundary representation – each face of the object is listed along with the vertices (their  $x, y, z$  co-ordinates that form the face. The vertices are listed in counter-clockwise order with the inside of the face on the left hand side as the vertices are traversed.

The low-level image processing techniques used in this thesis are relatively simple.



3 surfaces

on surface 0, there are altogether 4 vertices.

in counter-clockwise order they are:

(25 10 82) (46 40 241) (17 226 122) (5 202 19)

on surface 1, there are altogether 4 vertices.

in counter-clockwise order they are:

(48 228 84) (26 229 111) (46 40 241) (126 55 134)

on surface 2, there are altogether 6 vertices.

in counter-clockwise order they are:

(65 29 117) (112 40 62) (126 55 124) (46 40 241)  
 (25 10 82) (49 10 35)

Table 1.1: The output of the proposed system: a boundary representation. The numbers represent the  $x, y, z$  co-ordinates of the corresponding vertex.

However, more sophisticated ones can be used. Our main aim is to use the output of these low level techniques in extracting the 3-dimensional geometric information.

Our work is unique in the sense that it uses a novel way to get analytical description of roof edges. That is, it intersects (combines) surface equations from the region-based method in all possible ways and then validates (compares) the results with the roof edge map from the edge-based method to get the analytical descriptions for roof edges.

It then applies the above “combine and compare” idea into the process of getting vertex information. That is, vertex positions are derived by combining edge functions in all possible ways and then comparing the results with already known information on edges, such as its start and end point positions, to get rid of false vertex position declarations.

Further, the vertices are classified according to their pixel location and its relationship with the other pixels on the same surface. This is quite different from most existing classifications, such as junction dictionary, which is based on how lines intersect. According to the classification used in this thesis, as few as four types of vertices exist. False vertex declarations are removed by checking to see if its vertex type is allowed or not.

In the next few sections, we further discuss some important aspects of the thesis.



## **1.3 Why Assume a Polyhedral World?**

Most industrial objects are polyhedral; others that are not, can be approximated as polyhedra as far as planning collision-free paths is concerned. Most motion planning algorithms assume that the objects in the robot's environment are polyhedral. Furthermore, representation of polyhedral objects is fairly well understood in the solid modeling literature[27]. Hence, our choice of polyhedral world.

## **1.4 Why a Combined Approach?**

Most existing methods of extracting detailed boundary descriptions of polyhedral objects from range images are purely edge-based [32] [20]. This is especially true when a detailed scene description is required. But there are some problems with this approach. The following two subsections outline the pros and cons of the edge-based and region-based approaches to show where the hybrid idea comes from.

### **1.4.1 Pro and Cons of Edge-Based Methods**

The edge-based methods and the region-based methods, both provide us a way of segmentation. Segmentation algorithms are generally based on one of two basic properties: discontinuity and similarity. The edge-based methods belong to the first category. They partition an image based on abrupt changes in pixel values. For example, in a range image, this could mean sudden a change in distance – a jump edge. It could also mean a change in surface normal direction – a roof edge. In an

intensity image, this could mean a change in gray level, and so on. The main focus of the edge-based methods is usually the detection of isolated pixels, and detection of lines and edges in an image. Because this approach provides direct information on edges and vertices, it is easy to understand why many vision researchers prefer this approach to region-based ones when a boundary representation is desired. However to get an accurate edge description from this method is very difficult. For example, HT, one of the most commonly used technique of obtaining edge descriptions from edge maps, always has noise and false edge declarations. Inaccuracy in valid edge segment declarations are very common and sometimes almost inevitable. However in order to achieve a boundary representation, especially in order to get the information on which edge belongs to which surface and which edge separates which two regions, accuracy is very important. A small error here may lead to significantly different results later on and may cause wrong inference.

In order to get a more accurate boundary representation, many vision researchers have focused on improving the pure edge-based methods. However, the improvement has a foreseeable limitation because the edge-based approach alone sometimes cannot provide sufficient information to remove noise and obtain description about real edge and real edge alone.

#### **1.4.2 Pro and Cons of Region-Based Methods**

Region-based methods segment an image into regions of similar characteristics based on pixel properties. For example, the region can be separated according to its color, its intensity value, its depth value and so on. As you can see, information derived from this approach is not in an obvious edge-vertex form. Instead it is in a form

that describes a particular region property. Therefore, region-based methods are not considered straight forward to use for boundary representation. This is one of the main reasons why vision researchers seldom use this method to get boundary representations.

However, region-based information helps us get robust and accurate description of what is in the image, for example, surface (planar, in our case) equations. Many approaches derive their surface descriptions [20] purely from edge-based methods. It is not hard to imagine that a little error in the edge description may cause significant inaccuracy in the surface description derived later. The region-based approaches can give a more reliable result of things like surface equations. With this in mind, it seems natural to use information provided by region-based approach as a supplementary information to edge based one. Such descriptions also help the subsequent geometric analysis to extract valid vertices.

Summing up then, this thesis provides a hybrid approach to extract boundary representations of visible surfaces of a polyhedral object in a single range image. Such representations are to be used for path planning algorithms in a robotic work cell environment.

The remainder of this thesis is organized as follows. Chapter 2 presents a survey of recent literature that is related to this thesis work. Chapter 3 proposes the main algorithm combining the edge-based and region-based methods and the geometric analysis to extract boundary representations. It also discusses the implementation of this algorithm. Chapter 4 contains an evaluation of the algorithm and some experimental results. Chapter 5 concludes this thesis summarizing the results. It also points out some important issues raised by this thesis work and some possible

future research directions.

# CHAPTER 2

## Background And Literature Review

There are two broad sub-areas in our research work: (i) low-level range image processing techniques that extract geometric primitives such as jump edges, roof edges, and planar regions, and (ii) intermediate-level geometric analysis that extracts the boundary representation from these primitives. In this chapter, we review various approaches as applied to each sub-area.

## **2.1 Range Image Processing**

### **2.1.1 Edge Detection**

Edge detection for intensity image processing has been studied quite intensively. Many edge operators, such as Gaussian, Laplacian, Sobel and Canny [1], are available. In range images there are primarily two types of edges – jump edges and roof edges. A jump edge implies a discontinuity in depth, and a roof edge implies a discontinuity in surface orientation. Below, we review various approaches to both.

#### **2.1.1.1 Jump Edge**

Most of the edge operators that have been used for edge detection in intensity images, (since, in the abstract, they detect discontinuities in the two-dimensional image function) can be used to detect jump edges in the range images (refer to [36] for an example).

#### **2.1.1.2 Roof Edge**

Yokoya and Levine use surface discontinuity to detect roof edges in range images. They use the fact that there exist significant surface normal changes where roof edges occur. This method has been used in this thesis for its computational simplicity.

Approaches that are capable of detecting both roof and jump edges have been proposed [30]. Gaussian masks at different scales have been used to smooth the image. Derivatives are calculated using a set of characteristic contour masks, each

is sensitive to the detection of edges of different orientation and type. The edge map will be derived based on the above information. However, processing jump and roof edges separately gives us more information and sometimes can make the task easier.

### 2.1.2 Hough Transform

In this thesis, the Hough transform (HT) is used as a computationally efficient method for detecting lines (more general curves can also be detected) in images. It maps a line in the image space to a point in the Hough parameter space. Hough space could be made up of different parameter pairs, such as the slope-intercept and the angle-radius pairs. The angle-radius pair is preferable because they are both bounded while the slope-intercept parameters are both unbounded.

Although the HT is a very efficient technique to get line descriptions, it has some problems. One of these is the false peak point in the voting array that declares detected lines. To circumvent this problem, Herman[20], in his method, removes all edge pixels corresponding to a line detected in the voting array before looking for the next line in the image. Although the Hough voting is done in the  $\rho$  and  $\theta$  space, Herman chooses the two end points of the line as its symbolic representation. This makes the detected line information more explicit.

The HT technique used in this thesis uses this two end point form to represent the detected lines too. But instead of eliminating all voting edge pixels, a two step thresholding technique is used to circumvent some of the false line declarations in the voting array. This will help to eliminate noise and in the mean time maintain the ability to detect short edges.

### 2.1.3 Segmentation

Generally speaking, segmentation plays an important role in image analysis. Horn said in [21]:

Many image-analysis techniques are meant to be applied to regions of an image corresponding to single objects, rather than to the whole image. Because typically many surfaces in the environment are imaged together, the image must be divided up into regions corresponding to separate entities in the environment before such techniques can be applied.

This in a way shows the importance of segmentation in image analysis. In this thesis, the segmentation plays an important role. Each object should be separated into individual planar regions. This is necessary for extracting analytical descriptions of the plane surfaces for further geometrical analysis.

Various segmentation techniques have been studied. Some of these are: thresholding technique, histograms, split-and-merge technique, region growing and differential geometry. Most of these techniques are primarily used to extract low-level representations of surfaces for subsequent three-dimensional scene analysis. Their ability to partition a scene into planar, and sometimes even curved, surfaces is very useful for the surface segmentation part of this thesis.

The split method for segmentation begins with the entire image as the initial segment. It then successively splits each of its current segments into quarters if the segment is not homogeneous enough. Robertson [31] and Klinger [24] pioneered this research area. Because the boundaries produced by the split technique have a



quantization problem, the split-and-merge technique was proposed by Horowitz and Pavlidis [23]. This technique has been studied since then. Chen and Pavlidis(1980) suggest using statistical tests for uniformity, which requires the mean of the region and each of its quarters to be very similar. The Split-and-Merge segmentation technique has been used to segment aerial photographs by R.H. Laprade [25]. It uses the split-and-merge framework to perform efficient computations of the least squares approximations. The efficiency is due to the fact that the parameters needed to compute the LSF for the union of two regions can be computed by adding the corresponding parameters of each region. [25]

Oshima and Shirai [28] presented another approach to describe how to partition a scene into regions. The regions are classified into three classes: plane, curved and undefined. First, planes are fitted in a  $8 \times 8$  window. Regions where the error of approximation is small are used as seed regions and other regions are merged into them if they have similar plane equations. The program extends the curved regions into global regions by merging adjacent curved and undefined regions. The curved global region is approximated by a quadratic surface. Thus, the scene is described by plane regions and curved regions.

Using range image histograms to segment corresponding registered intensity images is proposed by Wong and Hayrapetian [35]. They separate a particular part in an intensity image according to the distance in the range image. They suggest that each pixel of every region should be within a certain range. This approach is useful for some specific applications, but it is difficult to extend it for general applications.

The histogram technique has also been used in Synthetic Aperture Radar image analysis [34]. In this paper[7], Duda, Nitzan and Barrett present a method that uses

registered range and intensity data to detect and extract regions that correspond to planar surfaces in a scene. A three step region clustering scheme is described in the paper. This scheme finds horizontal surface first, then look for vertical surfaces and finds the slanted regions last. In general, this method can partition a scene of multiple objects which have different polyhedral shapes into horizontal, vertical and slanted surface regions. The unlabeled regions correspond either to jump boundary pixels or to regions that are very small, non-planar or both. The completed partitioning results provided in that paper are very impressive. However, the region boundaries formed by adjacent pairs of regions detected by the method are not refined. In order to get a better boundary, they did some interactive experimentations, and found that the boundaries at which two adjacent planar regions intersect can be computed quite precisely from the intersection of the best fitted planes. This is a very interesting experiment. However the purpose of their research was just to check the segmentation results and it did not go further. Also the pair of surface equations that form the boundary are input interactively by human operators.

A mathematically rigorous approach to edge detection in range images is based on the use of differential geometry [3]. This method detects edges by calculating first and second partial derivatives over the entire image. It then calculates surface normals, the principal curvature and the Gaussian and mean curvature for each pixel based on the previous calculation. Theoretically, this method can only be applied where surfaces in the images are smoothly differentiable. Although range data may not always satisfy this criterion, good results have been obtained.

The above are only a few of many segmentation techniques available. Although each of them has problems, they can provide acceptable segmentation results for

many situations. The output of low-level segmentation, say planar regions, are then used for higher-level geometric analysis.

## 2.2 Range Image Geometric Analysis

The geometric analysis methods use the results of low-level segmentation to extract more symbolic representations. Since we are using a boundary representation for polyhedral objects, this implies extracting vertices, edges, and faces of the polyhedra. Which segmentation method to use largely depends on which method can provide the information needed for further processing in the most direct and explicit way. The edge-based approaches are obviously more straightforward than the region-based approaches for deriving a boundary representation of polyhedral objects. Thus most approaches to extracting boundary representation adopt the edge-based approaches to do the segmentation job. Sugihara pioneered this research area. He published a paper [32] in 1979 which constructed a permissible junction dictionary that represents the general knowledge about the physical nature of the three dimensional object world. The remarkable point of this work is that he applies this junction dictionary in the feature extraction stage in order to get the most reliable edge drawing. Sugihara's work is one of the few previous approaches that can provide a complete description of range images in terms of visible surfaces, edges and vertices. The other prominent work in this area was done by M. Herman in 1985 [insert ref]. Most of the other attempts at range image analysis did not result in as complete a description as Sugihara and Herman can provide. The following gives a brief account of what has been done by Sugihara and Herman, along with the pros and cons of their methods.

Sugihara proposes a method that uses a junction dictionary to guide the range data analysis. A polyhedral junction dictionary is formed based on the following four assumptions:

1. the scene is a collection of polyhedral bodies;
2. exactly three faces meet at each vertex;
3. accidental alignments do not occur in the scene, that is, neither two distinct edges nor two distinct vertices share a common position and no vertex lies on an edge of another object, and that
4. neither the light source nor the camera lies on any face planes (when they are extended), that is, a slight movement of the eye (i.e., the light source or the camera) does not cause a drastic change of the associated line drawing.

A main point of Sugihara's study is that knowledge of permissible junctions is used for "feature extraction". At each step of the analysis, the system consults the dictionary to predict positions, orientations and physical types of missing edges. By comparing a line drawing extracted from the range data with the list of permissible junctions, the system can tell whether the present drawing is consistent with physical reality or not. If not, the system is able to predict the locations and physical categories of missing edges. As Sugihara argues, by using his method, edge detection can be executed efficiently and reliably, since the junction dictionary suggests missing lines at each step of the analysis. This is true if the permissible junction dictionary can be constructed correctly beforehand.

A main limitation in Sugihara's work is that a very strong assumption is made, that is, all the vertices are trihedral. If this assumption is relaxed, the size of the

junction dictionary becomes too large. For instance, with the four assumptions and three conventions in the processing, the junction dictionary of two polyhedra objects has twenty entries, a very manageable size. Without the trihedral vertex restriction, the dictionary will become very large, and the efficiency of the dictionary guidance will disappear, and the method will be impractical. In order to circumvent this problem, Sugihara suggests two kinds of approaches: a probabilistic approach and an automatic dictionary-construction approach. However, these two approaches cannot really solve the problem. They only ease the tension under certain circumstances. The probabilistic approach requires knowing the probability of the occurrence of each junction type beforehand. However, it is not really clear how to obtain this probabilistic knowledge about the scene and the idea has not been developed further. The automatic dictionary-construction approach requires that the objects in the scene be restricted to a few prototypes, in which case a junction dictionary for those prototypes can be constructed automatically and be used in the system. This, we believe, is a very ad-hoc approach, and is not suitable in general.

An approach that is completely different in concept should be introduced to jump out of this very strong trihedral-vertex assumption. Martin Herman's work is one step towards this direction.

Herman's method can provide the description for most of the visible faces, edges, and vertices. He classifies edges into three kinds: occluding, convex, and concave. In his approach, occluding edge points are located where there is a discontinuity in depth or where there is a boundary between data and no data regions. Convex and concave edge points are found by calculating the curvature at each pixel over the entire image. If the curvature is a local maximum and exceeds a threshold, the

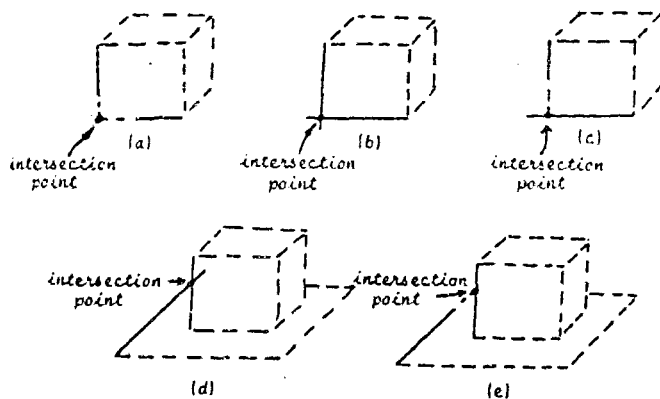


Figure 5: Connecting intersecting pairs of (extended) line segments.

Figure 2.1: Herman's approach

pixel is considered to be a concave edge point. On the other hand, if the curvature is a local minimum and exceeds a threshold, it is considered to be a convex edge point. Using these three definitions, Herman obtains three edge maps, namely the occluding edge map, the convex edge map, and the concave edge map. Next, Herman uses an improved version of HT to derive a symbolic description of all these edges respectively. In doing so, he implicitly assumes that all false edge declarations can be avoided using his improved HT technique and that any other noise in the edge map can be removed before forming junctions. Then he proposes an algorithm to form a vertex, which he refers to as a junction. This is done in three steps. First, Herman connects edge segments that are almost collinear and have close edge points. This is done by checking to see if the end points of the edges are within a threshold distance. If so, a junction is formed at the pixel midway between these end pixels. Second, intersecting pairs of (extended) edge segments are connected provided that the intersection points lie within a given threshold distance of the end pixels of the segments. Five examples given by Herman are shown in Figure 2.1. Herman explains:

In case 1, the intersection point lies outside the two segments(Fig. 5a).

Both are extended and a junction is formed. In case 2, the intersection point lies inside both segments (Fig. 5b). Both are shortened and a junction is formed. In case 3, the intersection point lies outside one segment but inside the other (Fig. 5c). The former is extended, the latter is shortened, and a junction is formed. In case 4, the intersection point lies inside both segments (Fig. 5d), but is beyond the threshold distance from each end point of one of the segments. The other segment is therefore shortened, but a junction is not formed connecting the two segments. Case 5 is the same as case 4, except the intersection point lies outside one segment but inside the other (Fig. 5e).

Herman argues that the thresholds used in the two steps just described above should be conservative and are only meant to connect edge segments that are quite close to each other. Otherwise, connections would be established between segments that should not be connected. Third, Herman assumes that if an edge segment has a dangling (i.e. unconnected) end pixel, it should be extended or shortened by a larger amount than the previously specified thresholds. Therefore, the second step is repeated with a much bigger threshold to obtain the right junction when this situation occurs.

The advantage of Herman's approach is that it makes relatively fewer assumptions about the object and it gives a relatively complete description of all visible faces, edges and vertices. However, one of the problems with Herman's approach is that it relies heavily on the accuracy of the HT to form the correct junction. The first two steps in Herman's junction-forming process require that the edge segmentation result should be very high so that the first threshold can be very conservative.

Another problem with Herman's approach is that the thresholds for the three step junction algorithm, particularly the threshold for the third step, is set with human interaction. In fact, it is very difficult, if at all possible, to extend Herman's junction forming theory to a non-interactive system. In a sense, this approach uses the human operator as a knowledge base to determine if a junction should be formed. A new way of defining the junction is needed in order to form all valid junctions and in the meantime minimize human interactions. The third problem with this approach is that whether the junctions shown in Figure 2.1 include all possible kinds of junctions of intersecting lines. This approach also suffers from the same problem that Sugihara's method does. The surface information is deduced indirectly from edge processing. Therefore, the reliability of the surface information relies heavily on the quality of edge processing.

In conclusion, Herman's approach provides a fairly complete description of the visible scene, but relies heavily on the quality of edge processing, particularly the HT, and human interaction to form a junction

This thesis provides a more generalized and data-driven approach. It uses information from both edge-based and region-based methods. Edge descriptions are derived directly from edge processing while surface information (analytical descriptions of planes that form the faces of the object) is obtained directly from raw image data using region-based approaches. Using region-based information, also facilitates a simpler geometric analysis to classify valid vertices in the scene. In the next chapter, we present our approach.



# CHAPTER 3

## The Description and Implementation of the Algorithm

### 3.1 Method Overview

As it is outlined before, our aim is to extract a B-rep type description of the visible surfaces of a polyhedral object from its range image. In order to get such a description of all the visible surfaces, a hierarchical scheme is suggested here. See Figure 3.1 for reference. We now briefly state the full algorithm (refer to Figure 3.2.) and then give details for each part in subsequent sections.

First, we extract a jump edge map and a roof edge map from the range image. Note that the term map implies that the representation is still at pixel level, i.e., each pixel is characterized as a valid or invalid edge pixel. Next, HT technique is used to extract symbolic jump edge segments. Independently, surface segmentation

techniques (based on surface normals) are then used to extract planar regions in the image. Least-squares-fit techniques are then used to determine analytical representation of each planar region. Each of these plane equations are then intersected (“combined”) to determine feasible roof edge segments. These roof edge segments are then validated against (“compared”) with the roof edge to determine valid roof edge segments. At this stage, we have all the edge segments (roof and jump) in the image. A very similar “combine and compare” method is used to extract valid vertices. A crucial point here is that vertex extraction is done on each plane separately. Having extracted the vertices, they are then grouped in a closed loop and ordered in a counterclockwise fashion (with simple geometric tests) to determine the final description of each face.

An overview of the implementation carried out in this thesis appears in Figure 3.3. The major processing steps outlined in this figure are described in each of the following sections. The input data is a range image obtained from the laser range finder with its corresponding binary flag image which indicates where regions of interests lie.

## **3.2 The Algorithm**

### **3.2.1 Jump Edge Map**

We define a jump edge as a discontinuity in the depth. Most edge operators used for intensity images can be used for detecting jump edges in range images. The quality of the edge map we will get from edge operators has a very important role in the

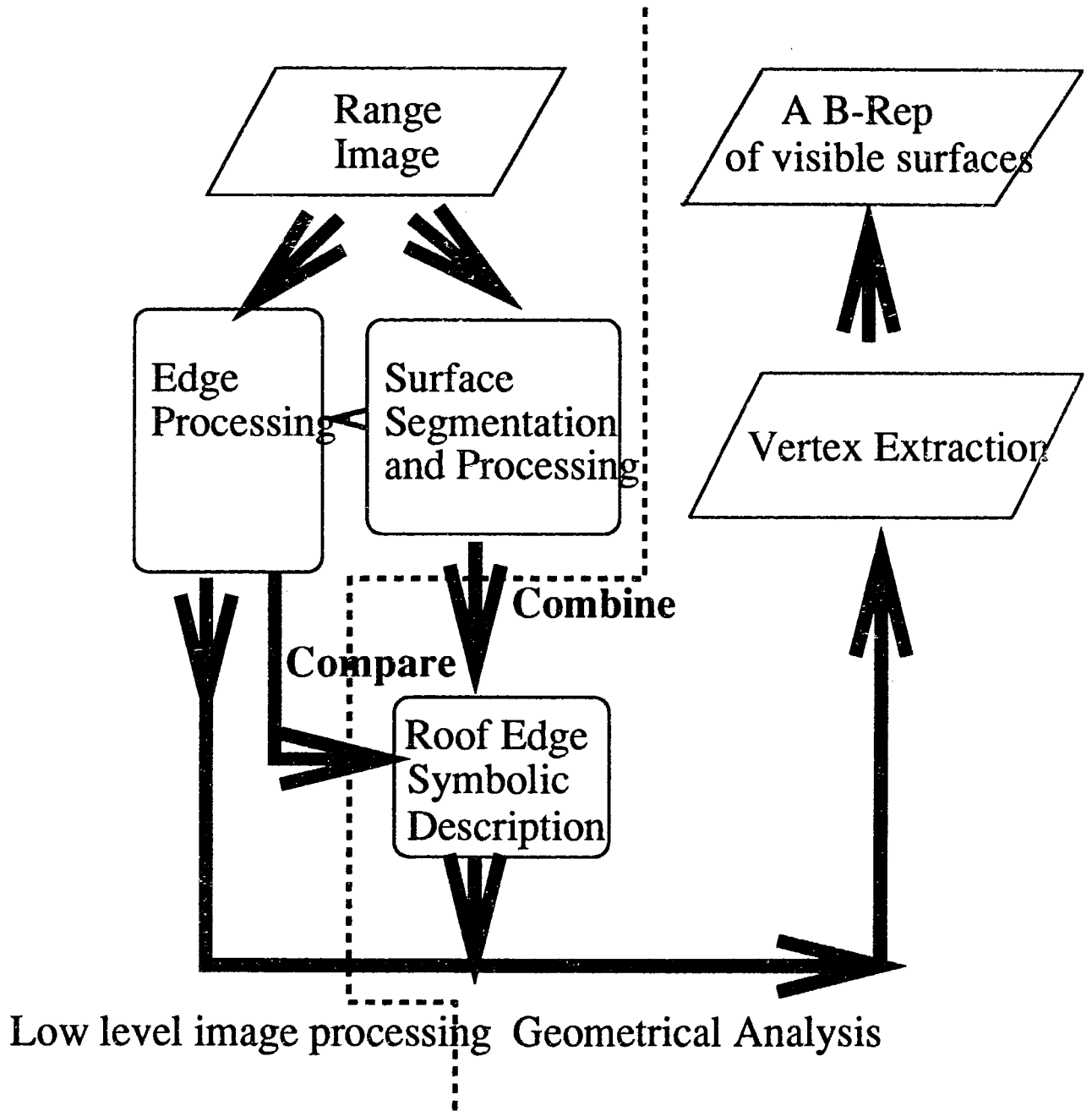


Figure 3.1: An Overview of Our Approach

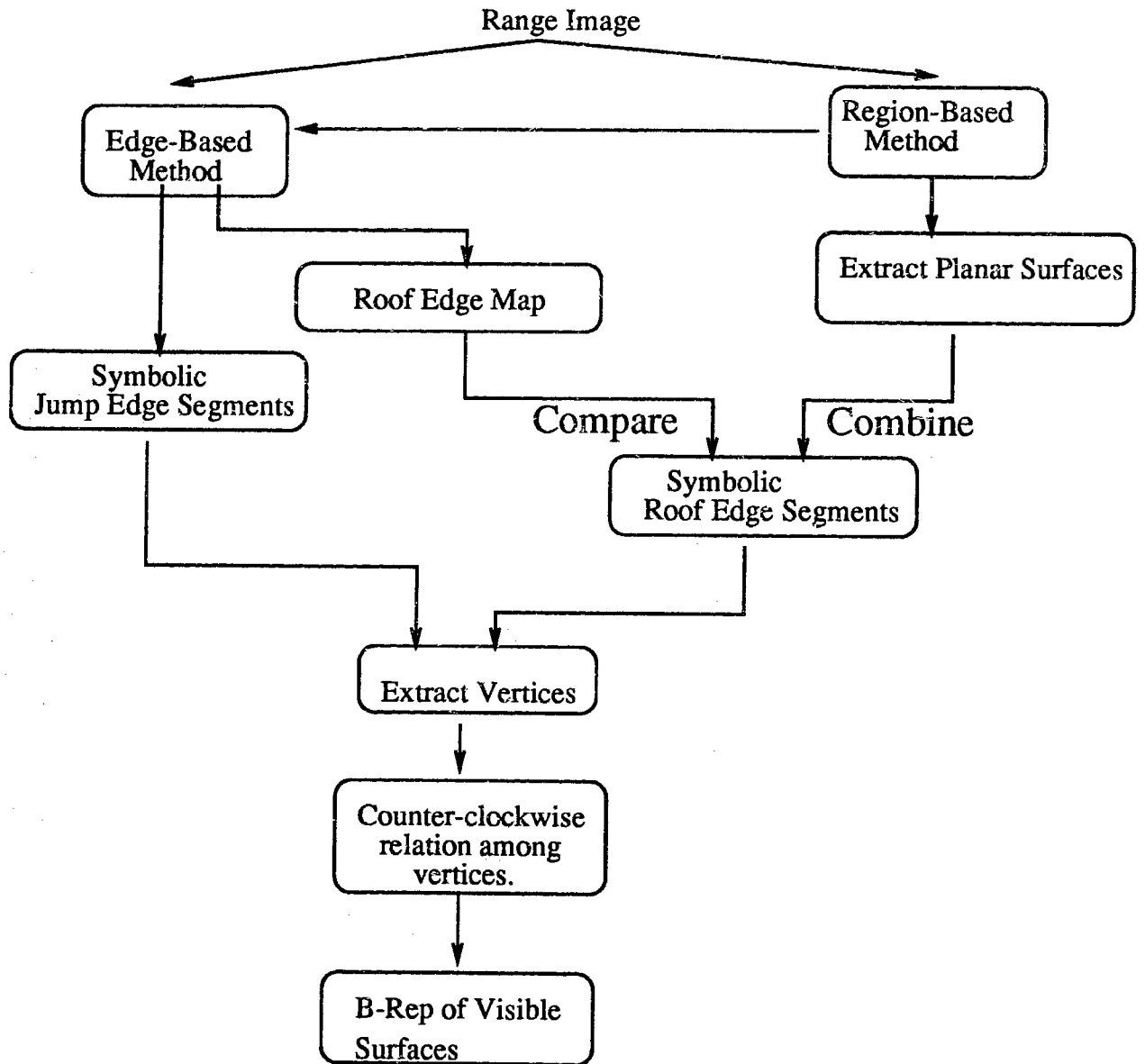


Figure 3.2: Overview of the proposed algorithm

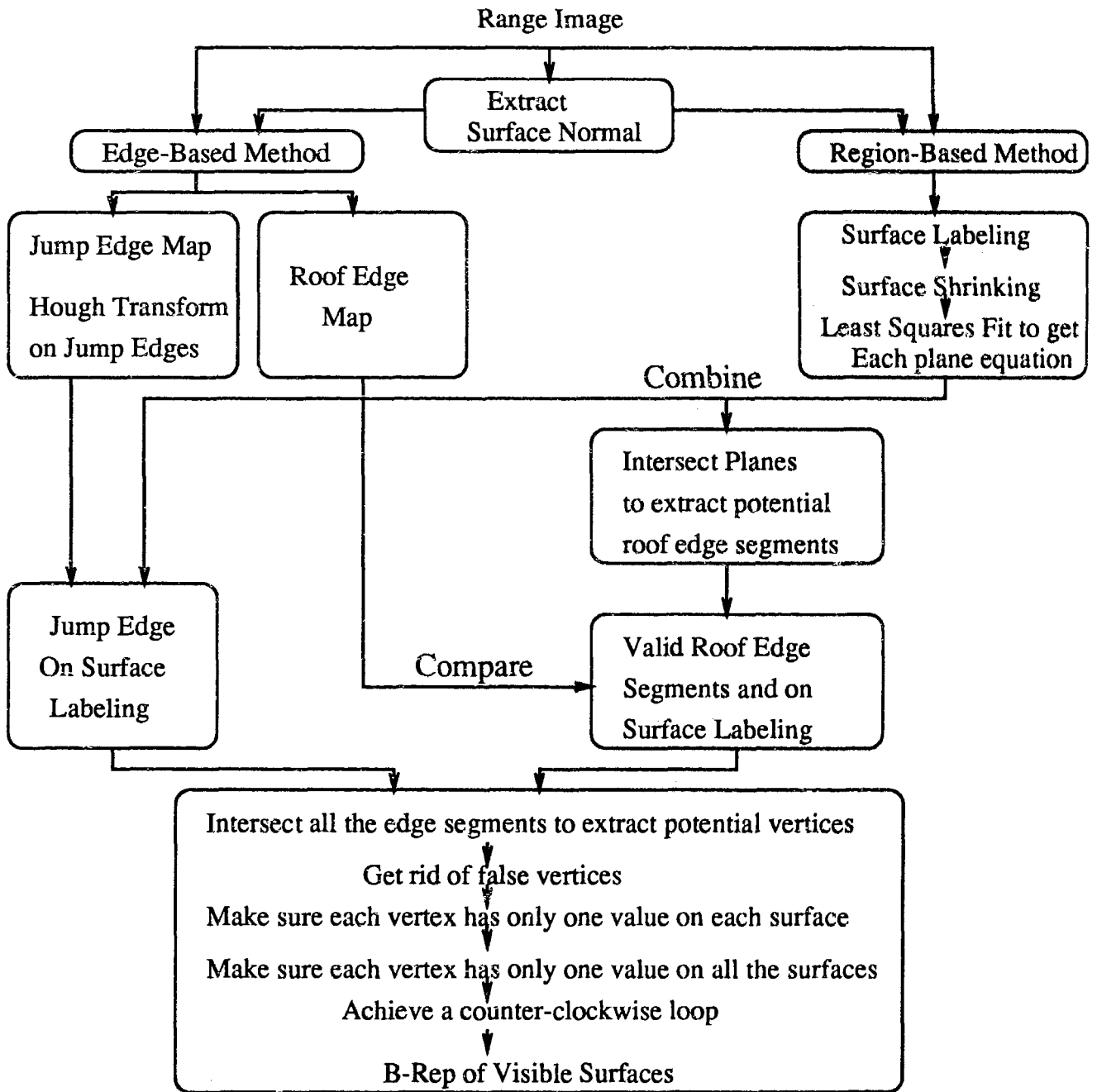


Figure 3.3: Overview of the implementation procedures

## Sobel Edge Operator

-1	0	1
-2	0	2
-1	0	1

1	2	1
0	0	0
-1	-2	-1

Figure 3.4: Edge Operators Used

later process of extracting a boundary representation. Hence, a good edge operator is essential. Different edge operators (Gaussian, Laplacian[5], Canny [4] and Sobel[1]) have been tried and the Sobel edge operator has been chosen among those primarily because of its simplicity and that at least for our test images, different edge operators gave very similar responses. Figure 3.4 shows the edge operators used in this thesis work.

The output of the two edge operators (shown in Figure 3.4), namely  $f_x$  and  $f_y$ , which are the convolution of  $F_x$  and  $F_y$  respectively, with the given image at a given pixel location, are used to calculate the edge magnitude and gradient. The magnitude indicates the strength of the edge, and the gradient gives the edge direction in degrees. The magnitude and gradient are calculated from the  $f_x$  and  $f_y$  using the following equations:

$$\text{magnitude} = \sqrt{f_x^2 + f_y^2} \quad (3.1)$$

$$\text{gradient} = \tan^{-1}\left(\frac{f_y}{f_x}\right) \quad (3.2)$$

```

typedef struct edge {
    int    starti[SIZE][SIZE];
    int    startj[SIZE][SIZE];
    int    endi[SIZE][SIZE];
    int    endj[SIZE][SIZE];
        } Edge: edge;

```

Table 3.1: The structure used in the HT to get the edge information

A jump edge is derived wherever the edge magnitude exceeds the desired threshold. The gradient of each jump edge pixel is stored for later processing (such as the HT on jump edges, section 3.2.2).

### 3.2.2 Hough Transform on Jump Edges

Jump edge pixels are grouped into edge segments using the HT technique. The original HT defines each edge segment by its  $\rho$  and  $\theta$  value. It only threshold the voting array once. The HT technique implemented here has two improvements over the original HT. First, the HT implemented here provides the start and end points, in addition to the  $\rho$  and  $\theta$  values. This definition of two points is more explicit and easier to use when try to find which surface each edge is in. In order to get the positions of the start and end points, the voting for the accumulator array is traced using an array structure as shown in Table 3.1. The subscripts of the arrays in this structure store the value of  $\rho$  and  $\theta$  which describes a line that has the start and end pixels stored in this structure. The algorithm for keeping track of the start and end points is shown in Table 3.2.

```

for(i=0; i<rows; i++)
for(j=0; j<cols; j++)
{
    If this is the first vote for a particular rho and theta,
        put the current i and j value to edge.starti[rho][theta] and
        edge.startj[rho][theta] respectively.
    If this is not the first vote for a particular rho and theta,
        put the current i and j value to edge.endi[rho][theta] and
        edge.endj[rho][theta] respectively.
}

```

Table 3.2: The pseudo-code for the algorithm for tracing in the HT

The other improvement is done to remove noise. The original HT can be expected to have some problem with noise. In order to remove the noise as much as possible and still be able to detect short existing edges, a two-step-threshold method is used in this thesis, as described in Table 3.3. With this two step threshold technique, we can eliminate much of the noise while maintaining the ability to detect the short edge segments.

Also, gradient information of edge pixels is used to guide the HT voting [1]. For a jump edge, the gradient information is calculated according to Equation 3.2. Using this gradient information greatly reduces the computational burden for the HT.



```

For the binary jump edge map
for(i=0; i<rows; i++)
for(j=0; j<cols; j++)
{
    Do the HT voting in the rho and theta space.
}

for(rho=0; rho<2*sqrt(maxi*maxi+maxj*maxj); rho++)
for(theta=0; theta<360; theta=theta+10)
{
    if(vote[rho][theta] > threshold1)
    {
        vote2[rho][theta] = the sum of votes in the neighborhood
        of (rho,theta) in the vote array;
    }
}

for(rho=0; rho<2*sqrt(maxi*maxi+maxj*maxj); rho++)
for(theta=0; theta<360; theta=theta+10)
{
    if(vote2[rho][theta] > threshold2)
    {
        claim the edge as a possibly valid edge;
    }
}

```

Table 3.3: The pseudo-code for the HT used in this thesis

P		
1	0	-1
1	0	-1
1	0	-1

Q		
1	1	1
0	0	0
-1	-1	-1

Figure 3.5: Gradient Operator Used

### 3.2.3 Surface Normal Calculation

Information on the surface normal at each pixel is useful for roof edge detection and region labeling. The surface normal has been calculated using the following two methods.

- Gradient method [21]:

The unit surface normal is :

$$\hat{\mathbf{n}} = \frac{\mathbf{n}}{|\mathbf{n}|} = \frac{(-p, -q, 1)^T}{\sqrt{1 + p^2 + q^2}} \quad (3.3)$$

where  $p$  and  $q$  are the gradient of the surface in the  $x$ - and  $y$ -directions respectively. Figure 3.5 shows the  $x$  and  $y$  gradient operator used in this thesis work.

- Least Squares Fit method:

This method fits a surface on  $z = ax + by + c$  over each pixel's neighborhood, giving the parameter  $a$ ,  $b$  and  $c$ . The surface normal will be  $(-a, -b, 1)$ .

The least squares plane is calculated as follows:

Assume:

$$A = \sum x_i^2 - \frac{\sum x_i \sum x_i}{n} \quad (3.4)$$

$$B = \sum x_i y_i - \frac{\sum x_i \sum y_i}{n} \quad (3.5)$$

$$D = \frac{\sum x_i \sum z_i}{n} - \sum x_i z_i \quad (3.6)$$

$$E = \sum y_i^2 - \frac{\sum y_i \sum y_i}{n} \quad (3.7)$$

$$F = \frac{\sum y_i \sum z_i}{n} - \sum y_i z_i \quad (3.8)$$

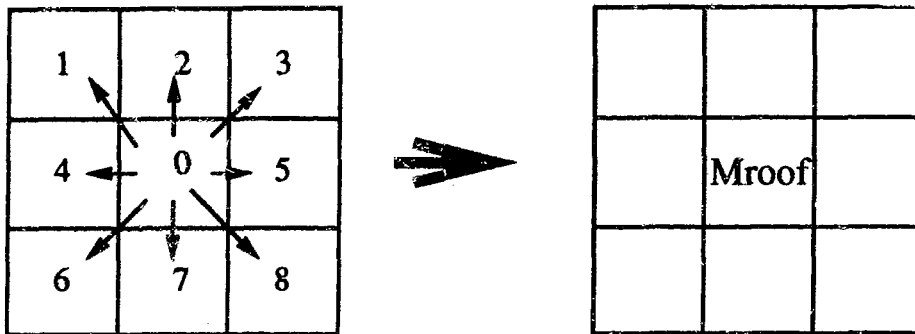
Then

$$a = -\frac{Eb + F}{B} = -\frac{E}{B} \times \frac{AF - DB}{B^2 - AE} - \frac{F}{B} = \frac{DE - BF}{B^2 - AE} \quad (3.9)$$

$$b = \frac{AF - DB}{B^2 - AE} \quad (3.10)$$

$$c = \frac{\sum z_i - a \sum x_i - b \sum y_i}{n} \quad (3.11)$$

The noise type of an image dictates which one of the above two methods to use. In practice, the method that provides the best roof edge map will be used to calculate the surface normal information. The quality of the roof edge map is checked interactively. A good surface normal information is essential for later processing. Further exploration into different methods to calculate surface normal at each pixel position may improve the algorithm's robustness.



- 1 Calculate the angular difference between the surface normal vector at position 0 and the neighboring position in the window.
- 2 Store the maximum value from step 1 at position 0.

Figure 3.6: Roof Edge Operator Used

### 3.2.4 Roof Edge Map

A roof edge is formed by two visible surfaces appearing in the image, i.e., it is a discontinuity in surface orientation. In order to get the roof edge map, surface normal discontinuities are detected in the image [36]. In our case, the roof edge magnitude, denoted by  $M_{roof}$ , is computed as the maximum angular difference between adjacent unit surface normals. This is formally given as:

$$M_{roof}(x, y) = \max\{\cos^{-1}(\mathbf{n}(x, y) \cdot \mathbf{n}(x + k, y + l)) : -1 \leq k, l \leq 1\} \quad (3.12)$$

where  $\mathbf{n}$  denotes a unit surface normal obtained from the previous section.

The actual calculation is illustrated in Figure 3.6. In this implementation, the mask size is adjustable. One problem found in this implementation is that it may detect some jump edges while it detects roof edges. The reason for this is that background pixels may have a different orientation from object pixels. There exists

discontinuity between the background surface normal and object surface normal. In order to remove this kind of false roof edge a comparison is made between the binary jump edge map and the initial binary roof edge map. At a position where there is a roof edge pixel, the jump edge map is checked to see if there is a jump edge pixel at that position. If a jump edge pixel is found nearby, then the corresponding roof edge pixel is removed. This is illustrated in Figure 3.7.

After all false roof edges are removed, the valid roof edge map is derived.

### **3.2.5 Surface Segmentation and Description**

Previous sections mainly discuss issues on the edge-based methods. This section begins by discussing the region-based method. We use a least-square-fit approach to get the analytical equations for the planes corresponding to the faces of the object.

Before performing the least-squares-fit, the object must be separated into different surface patches. It is part of the image segmentation technique. There are various ways to segment an image into respective regions of interest as outlined in Chapter 2. In this research the following straight forward method has been chosen for segmentation. The main principle guiding this process in this thesis is to group pixels according to their surface normal vectors. Pixels in the same plane should have very similar surface normal vectors. And so, the angular difference between them should be less than the threshold. The actual procedures are described in the following subsections.

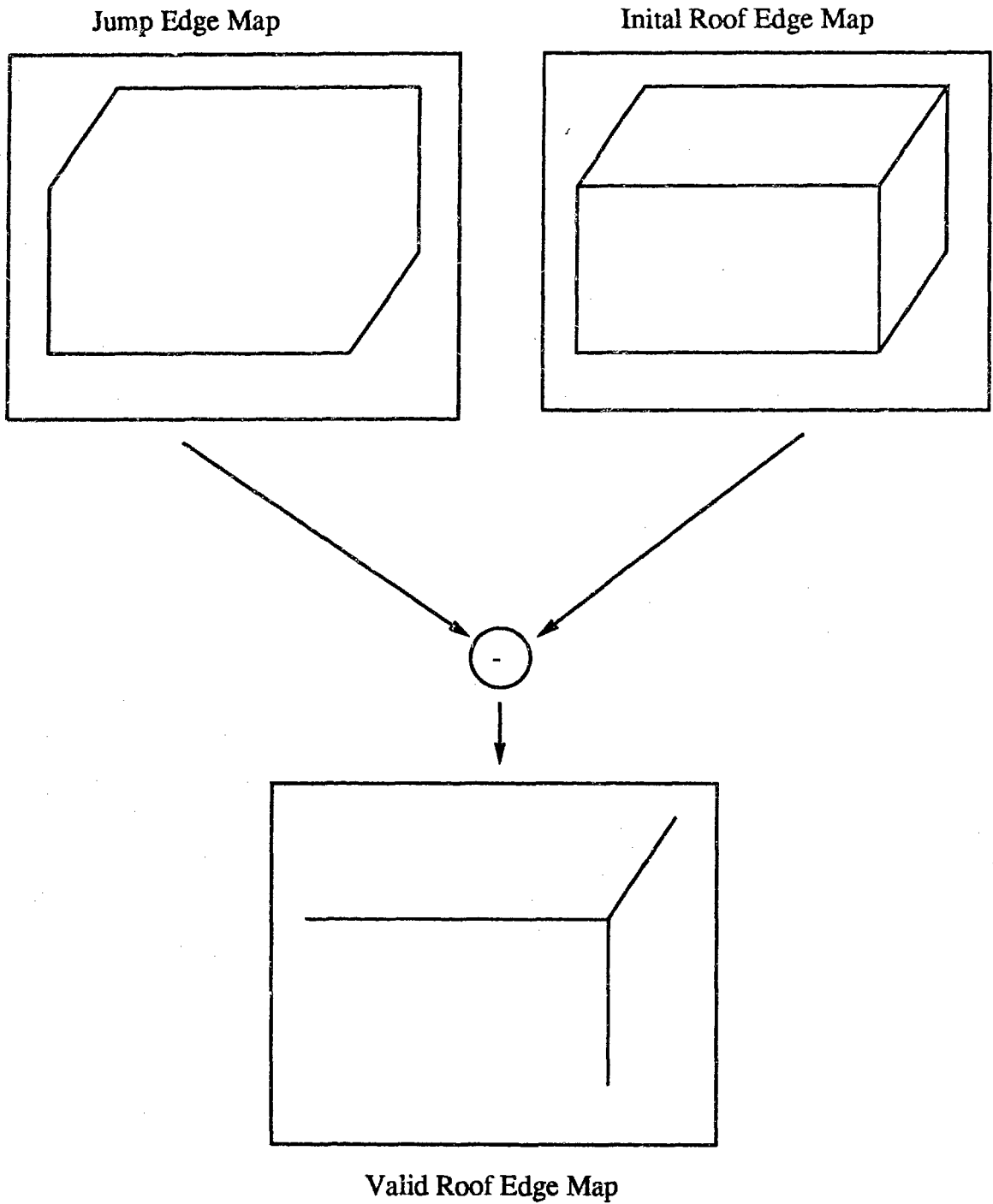


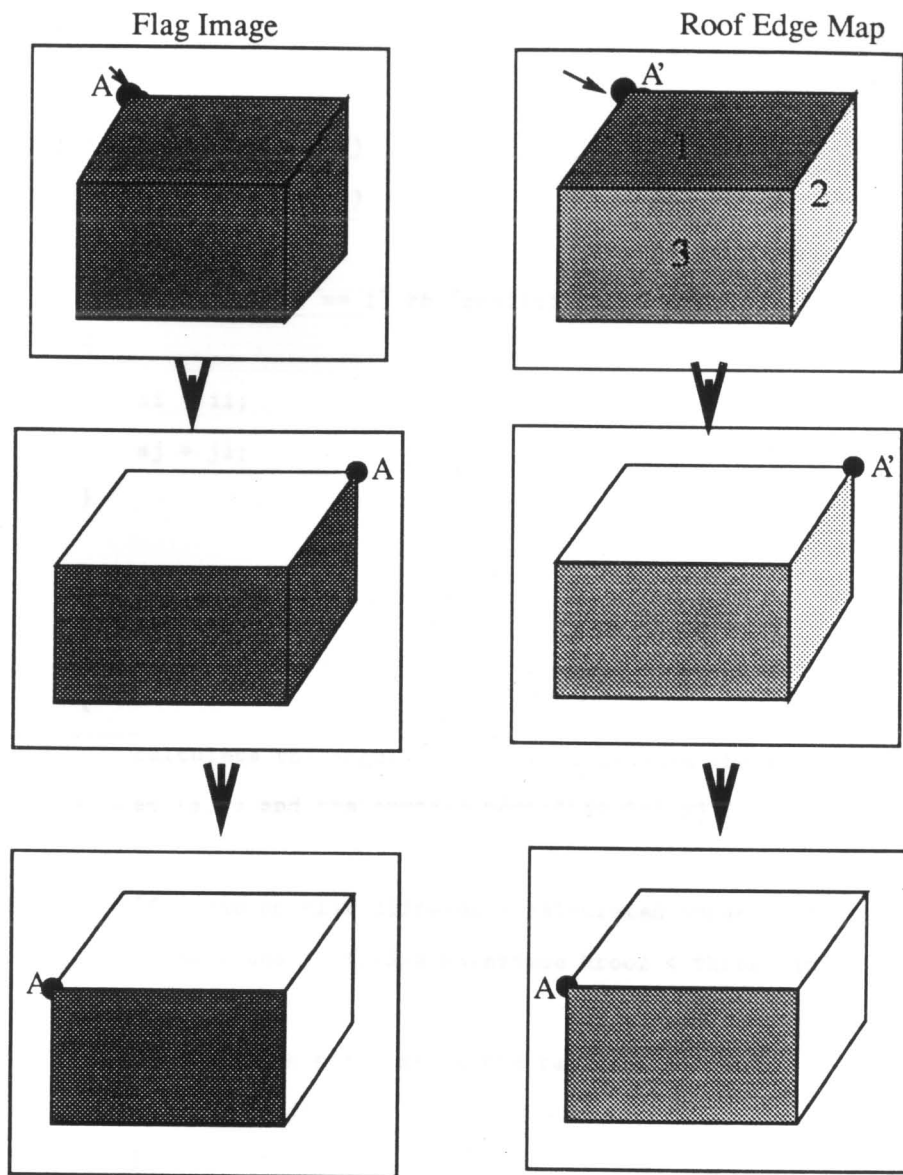
Figure 3.7: Remove false roof edges

### 3.2.5.1 Surface Segmentation

In order to partition an object into individual regions, the following steps must be taken:

1. Choose the first non-edge pixel position of one surface. The surface normal vector at this pixel position will then be called *the reference normal vector*. This reference normal vector is selected by scanning the current flag image and the original roof edge map. The first pixel position whose roof edge magnitude is less than the threshold and whose flag image value is one will be the reference normal position. Thus, the surface normal vector at that position is considered the current reference normal vector (See Figure 3.8).
2. After picking up a reference normal vector, scan the image from the beginning. Calculate the angular difference between the surface normal at the pixel being scanned and the current reference normal vector. If both the angular difference and the roof edge magnitude  $M_{roof}$  are less than the threshold, then this pixel belongs to this region.
3. If the pixel belongs to the region, this pixel will be removed from the current flag image and the roof edge map.
4. Repeat Step Two and Three until the number of pixels labeled in one row of the image exceed a threshold.
5. If there are still unlabeled pixels in any of the regions, then repeat the entire procedure starting with step one.

The pseudo code for the above procedure is shown in Table 3.4. After these steps



- 1 A and A' are at the same pixel location in the flag image and the roof edge map.
- 2 Surface normal at position A is chosen as the reference surface normal.
- 3 Calculate the angular difference between the surface normal at the current scanning position and the current reference surface normal.
- 4 If both the calculated angular difference and the roof edge magnitude are less than the threshold, the pixel belongs to the region.
- 5 If the pixel belongs to the region, set the value to zero in both of the images.

Figure 3.8: Selecting the reference surface normal



```

for(i1=0; i1<rows; i1++)
for(j1=0; j1<cols; j1++)
{
    if((flag[i1][j1] == 1) && (roof[i1][j1]<threshold1))
    {
        si = i1;
        sj = j1;
    }

    for(i=0; i<rows; i++)
    for(j=0; j<cols; j++)
    {
        calculate the angular difference between the surface normal
        at (i,j) and the surface normal at (si,sj).

        If ((the angular difference calculated above<threshold2)
            && ( the roof edge magnitude Mroof < threshold2))
        {
            label the pixel as in the region;
            flag[i][j] = 0;
        }
    }
}
}

```

Table 3.4: The pseudo-code for surface segmentation

are complete, planar regions in the image are known and have been labeled.

### 3.2.5.2 Region Shrinking

After the planer regions in the image have been labeled, there is likely to be some noise in the labeled regions, especially at the boundaries. In order to remove the noise, we shrink the labeled regions using the method illustrated in Figure 3.9. The procedures are:

1. Put the shrink operator over the flag image. If the sum of the values stored in the flag image at the shrink operator position is less than nine, then the pixel position corresponding to the center of the shrink operator is a boundary pixel.
2. If a pixel is found to be a boundary pixel, then remove it.
3. Repeat the above procedure until an acceptable quality of the image is achieved. Normally, around four to five iterations are sufficient.

### 3.2.5.3 Generating Surface Equations

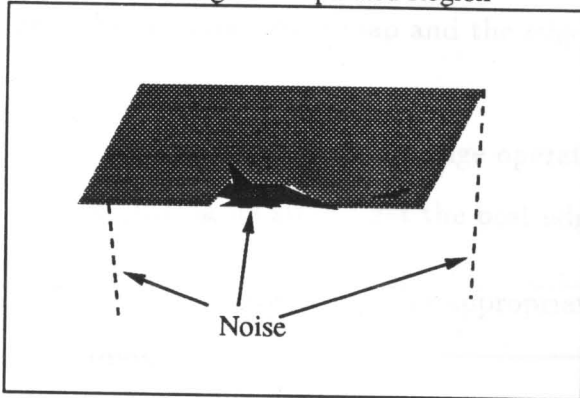
In order to get the equation that describes each plane, a least-squares-fit is done separately on each labeled region. This is done by performing a surface fitting of  $z = ax + by + c$  on each separated shrunk regions respectively to get the  $a$ ,  $b$  and  $c$  values. The surface normal will be  $(-a, -b, 1)$ . For details, refer to section 3.2.3 on surface normal calculation. Thus, the surface functions for each plane are now derived.

Shrink Operator



1	1	1
1	1	1
1	1	1

An Image for Separated Region



Its Corresponding Flag Image

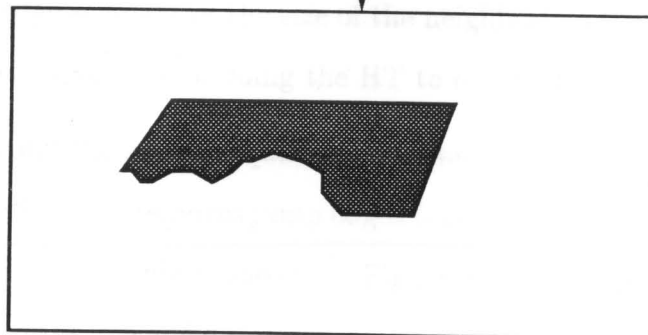
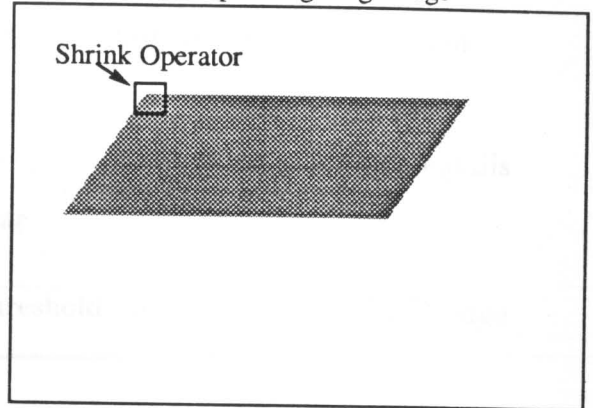


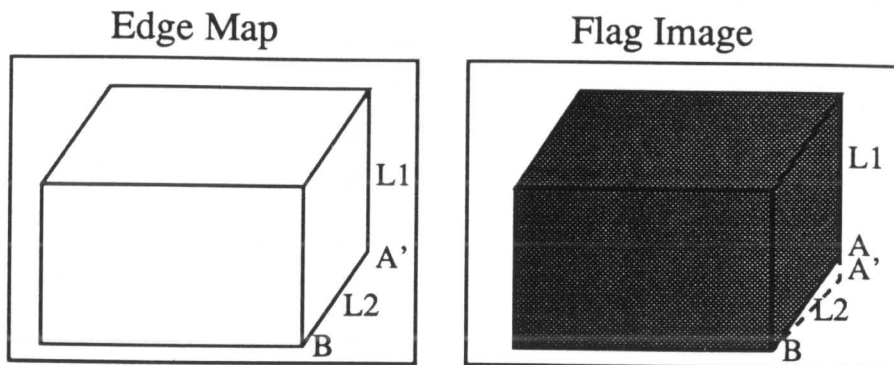
Figure 3.9: Illustration: shrink the region

### 3.2.6 Jump Edge Labeling

The jump edges must now be examined to determine which surface the detected jump edges belong to. The start and end points of each jump edge are analyzed to see if they both lie on the certain plane whose equation is derived in the previous section. Normally, jump edges defined in this thesis can only be located on one visible surface. Therefore only the plane that gives the minimum error is chosen as the plane that the jump edge belongs to. The following has been done in order to get a better jump edge map and the edges' corresponding analytical description:

1. Interactively adjust the edge operator to get the best edge map. This entails adjusting its size to get the best edge map.
2. Choose interactively the appropriate threshold to get a better binary edge map.
3. Use the flag image to ensure that all edge pixels are valid, and in doing so, background noise is eliminated. (See Figure 3.10)
4. Adjust parameters (parameters such as the minimum number of votes to declare an edge segment or the size of the neighborhood in which to group votes, etc.) interactively when doing the HT to eliminate false edge declarations.

After getting the surface equations, another step of noise removal can be performed. This step removes jump edges falsely made up of pixels from different surfaces. An example is shown in Figure 3.11. In this case, a declared edge  $A'B$  cannot be labeled onto any one of the known faces, so it is removed as noise.



Detected edges L1 and L2 intersect at vertex A'. A' is not inside the region of interest shown by the flag image.

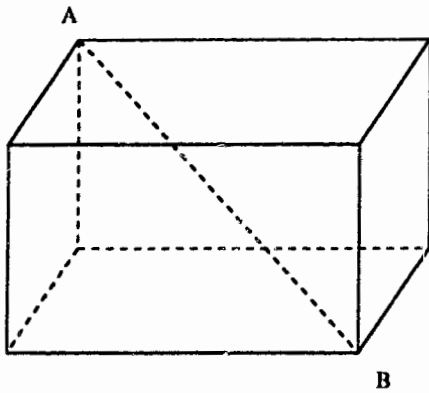
Figure 3.10: Illustration

5. In order to remove false edge declarations that are made up of pixels from different planes, several pixels claimed in the symbolic representation of jump edges are checked to see if this declaration of the jump edge is valid. If the pixels checked belong to the same plane, the jump edge is considered valid. For efficiency, we check each *mth* point along the line.

After this has been done, we assume that all false jump edge declarations have been removed.

### 3.2.7 Roof Edge Segmentation and Labeling

At this stage, we want to get a symbolic description of all the roof edges in the roof edge map. The typical method to do this is the HT. The weakness of the HT is the noise it generates, such as false edge declaration, and its inaccurate description. This can happen even when the edge map is very clean. Here the roof edge map



False edge declarations, such as edge AB shown in this figure, can be removed by checking to see if Edge AB is in any of the known surfaces.

Figure 3.11: Illustration

itself has lots of noise too. This makes it even more difficult to get an accurate analytical description of all roof edges from the HT method. However, in order to achieve a boundary representation, a good edge description is essential. Therefore, a novel approach is proposed and used here in order to get a better result. (The pseudo code for this approach is shown in Table 3.5.)

```

num = number of plane functions;
num1 = num*(num-1)/2

for(k=0; k<num1; k++)
{
    /* assume the surface equation is: *
    * z = ax+by+c;                    */
    a1 = equation(0).a;
    b1 = equation(0).b;
    c1 = equation(0).c;
    a2 = equation(1).a;

```

```

b2 = equation(1).b;
c2 = equation(1).c;

s = 1;
m = 20;
for(i=0; i<rows; i++)
for(j=0; j<cols; j++)
{
    if((abs(z-(a1*j+b1*i+c1))<threshold) &&
        (abs(z-(a2*j+b2*i+c2))<threshold))
    {
        if(s)
        {
            edge[i].starti = i;
            edge[i].startj = j;
            s = 0;
        }
        else
        {
            edge[i].endi = i;
            edge[i].endj = j;
        }

        if((m>0) && (mod(m/2)==0))
        {
            edge[i].midi = i;
            edge[i].midj = j;
            m = m - 1;
        }
    }
}

```

}  
 }

Table 3.5: The pseudo-code for surface segmentation

The approach is as follows.

First, combine these plane functions into groups of two in all possible ways. If there are  $n$  plane equations, then there will be  $C_2^n$  possible combinations. Each combination represents a potential roof edge segment.

Next, find all pixels in the image that belong to both of the two planes using the structure shown in Table 3.6. This is done by finding pixels in the image that can be identified as being on both of the planes that form the roof edge within a certain error tolerance. During this process, the start and end points of the edge are recorded as in the case of jump edges. Another representative set of pixel positions (ten, in our case) are also recorded. The analytical description of the edge is stored in the  $\rho$ - $\theta$  parameterization. The edge's corresponding  $\rho$  and  $\theta$  values are derived using the following method:

$$\theta = \tan^{-1} \frac{y_1 - y_2}{x_1 - x_2} \quad (3.13)$$

$\rho$  is the distance between the intersection of the line through origin that has the  $\theta$  value calculated above and the edge function to the origin.

Finally, check the binary roof edge map to see if there are physical roof edge pixels that match the edges claimed above. Only those claimed roof edge pixels that can find their counterpart in the binary roof edge map are claimed as real roof



```

typedef struct fedge {
    int    rho;
    int    theta;
    int    starti;
    int    startj;
    int    endi;
    int    endj;
    int    midi[10];
    int    midj[10];
    int    pixel;      /* the number of mid pixels   *
                       * recorded                       */
    int    label[3];  /* which surface its on   */
    int    flag;      /* if flag=0; it is not a *
                       * valid edge.           */
    int    midflag[10]; /* 0 means that pixel is not *
                       * an edge pixel.       */
}

```

Table 3.6: The structure used to get the roof edge information.

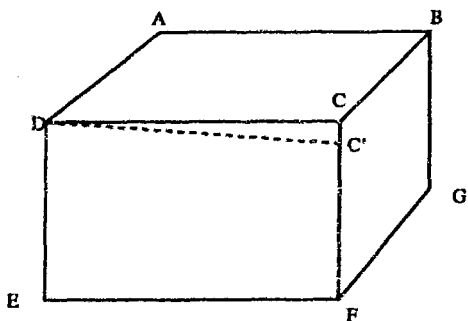


Figure 3.12: Illustration

edges. If no correspondence in the roof edge map is found, that edge will be removed as a noise. The accuracy of this method can be enhanced if we check more pixels on the edge.

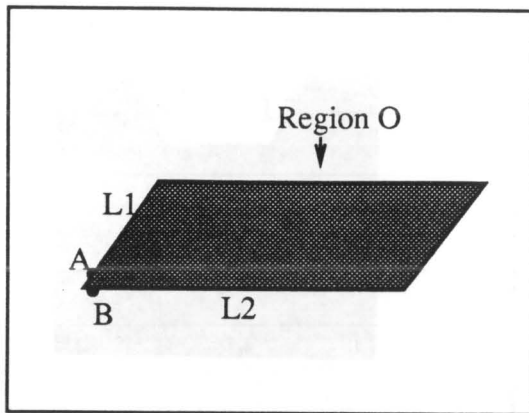
The reason that this method is preferable to the Hough Transform is that very small inaccuracy of roof edge symbolic description derived from the HT will result in very big error when trying to know which plane they are on. See for example in the Figure 3.12. In this case, let us assume the HT is used to get the information of roof edges. If there is a small inaccuracy occurs during the HT, and instead of an accurate result of the description of edge  $DC$ , we get the description of  $DC'$ . This is very likely to happen during the HT. Edge  $DC$  is a common edge between Surface  $DABC$  and Surface  $EFCD$ . Edge  $DC'$  is an edge that lies solely in the Surface  $DCFE$ . Then if we try to label Edge  $DC'$  onto the surfaces this amount of inaccuracy may be intolerable. However, in our case, the symbolic edge description is derived from the intersection of two planar equations, which are quite robust, since they have been derived using a relative large set of pixels in the region using least-squares-fit. Furthermore, the error can also be controlled within an acceptable range by interactively setting the threshold for roof edge fitting.

Now we have got symbolic descriptions of all roof edges and which two planes form these roof edges, and jump edge descriptions. For further geometric analysis, we will assume that the description is accurate. This means that all the edges are in the description and all false edges have been removed.

### 3.2.8 Extracting Vertices

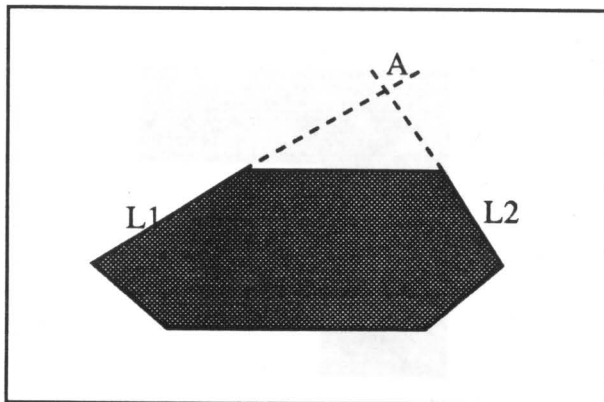
Next step in our method is to extract vertices explicitly. The following algorithm extracts vertices.

1. Combine the edges in all possible combinations to get all potential vertices. Therefore, if there is  $n$  edges, the number of all possible combinations should be  $C_2^n$ .
2. After all potential vertex positions are found, we will classify each one of them into the following five different categories:
  - valid vertices I that are obvious. This refers to the situation shown in Figure 3.13. In this case, the already known vertex position, from the edge processing, namely the stored start and end points, of the edges that form that vertex are within a given threshold.
  - false vertices that lie completely outside a region. See Figure 3.14 for an example.
  - false vertices that lie completely inside the region. See Figure 3.15 for an example.
  - false vertices that lie on the edges. See Figure 3.16 for an example.



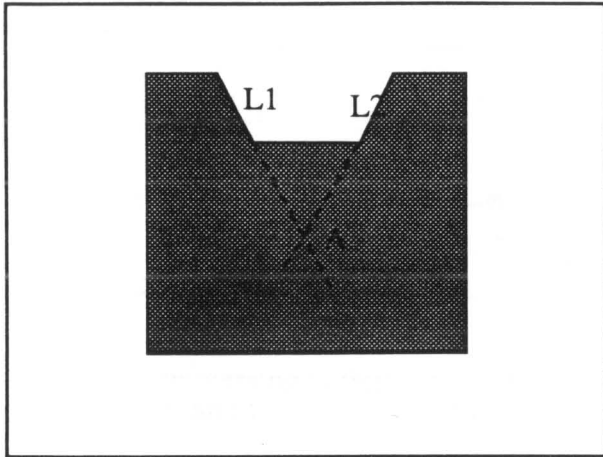
Vertex A of L1 and Vertex B of L2 are within an acceptable distance. Therefore, they are both labeled as valid vertices.

Figure 3.13: Illustration: a real vertex that are obvious



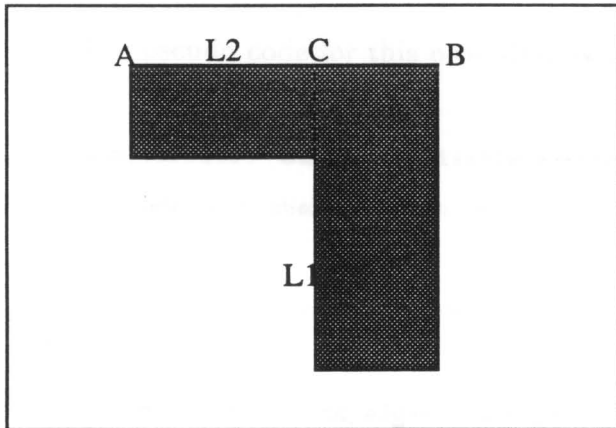
Line L1 and Line L2 intersect at A. But A is outside the region of interest. Therefore, vertex A is labeled as invalid.

Figure 3.14: Illustration: a false vertex that is outside a region



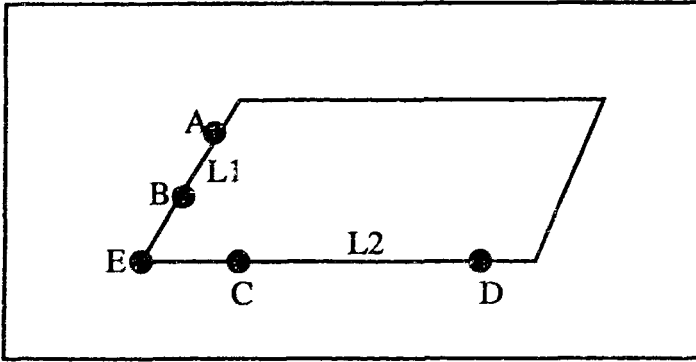
Line L1 and L2 intersect at A which is a pixel totally inside the region of interest.

Figure 3.15: Illustration: a false vertex that is inside the region



Line L1 and L2 intersect at C. C is in between vertex A and B. Therefore C is not a valid vertex.

Figure 3.16: Illustration: a false vertex that is on the edge



- 1 Edge processing indicates that edge L1 has vertices A and B while edge L2 has vertices C and D.
2. The relation between the calculated vertex E and the above claimed vertices A, B, C, D does not belong to any of the vertex categories previously shown. So E is claimed as a valid vertex.

Figure 3.17: Illustration: an example of a valid vertex II.

- valid vertices II. This refers to any vertex that does not fall under the previous four categories. Refer to Figure 3.17 for illustration.

The pseudo code for this procedure is shown in Table 3.7.

```

num_surface = number of visible surfaces;
num_edge = number of edges on each surface;

for(i=0; i<num_surface; i++)
{
    num_vertex = num_edge*(num_edge)/2;
    for(j=0; j<num_vertex; j++)
    {
        vertex(j) is formed by edge L1, L2;
        check to see if the start and point of these two
        edges are within a neighborhood,
    }
}

```

```

        if so,
            the vertex(j) is valid;
        else
            vertex(j) remains being labeled invalid;
    }
}
/* -----*/

the calculated vertex position is (si,sj);
It is formed by edge L1 and L2;

i1 = si-1;
j1 = sj-1;
i2 = si+1;
i2 = sj+1;
if(i1<0)
    i1 = 0;
if(j1<0)
    j1 = 0;
if(i2>(rows-1))
    i2 = rows;
if(j2>(cols-1))
    j2 = cols;

sum = 0;
for(i=i1; i<i2+1; i++)
for(j=j1; j<j2+1; j++)
    sum = sum+flag[i][j]; /* the flag array shows which *
                           * surface is being processed. */

```

```

if(sum == 0)
    vertex(si, sj) is outside the region, and therefore
    invalid;
else if(sum == 9)
    vertex(si, sj) is inside the region, and therefore
    invalid;
else if (0<sum<9)
    vertex(si, sj) is on edges;

/* -----*/
if( vertex(si, sj) is on edge)
{
    if(vertex(si,sj) is in between edge L1's start and end
    point, vertex(si,sj) is invalid;

    else if(vertex(si,sj) is in between edge L2's start and
    end points, vertex(si, sj) is invalid;

    else if the above two are not true, then
    {
        calculate some pixel positions that are on Line L1
        between the vertex(si,sj) and the start or end point
        of L1 depends on whichever is closer to (si,sj);

        check with the binary edge map to see if these
        calculated pixels have corresponding real edge pixels
        in the edge map,
        if not, (si,sj) is invalid;
        if yes,
        {

```



```

        calculate some pixel positions that are on Line L2
        between the vertex(si,sj) and the start or end point
        of L2 depends on whichever is closer to (si,sj);

        check with the binary edge map to see if these
        calculated pixels have corresponding real edge pixels
        in the edge map,
        if not, (si,sj) is invalid;
        Otherwise, (si,sj) is valid;
    }
}
}

```

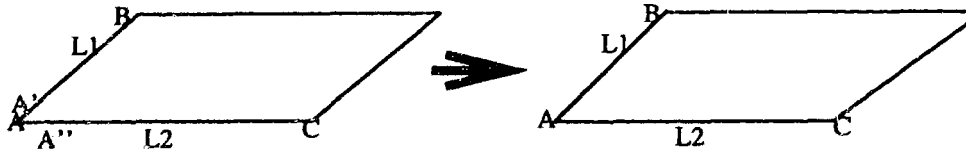
Table 3.7: The pseudo code for vertex confirmation

Thus, we have determined all the valid vertex locations.

### 3.2.9 Closed Loop Description

Before deriving this close loop description, we have to address one point. Since the vertices are extracted plane by plane, there are some minor differences among the derived vertex coordinates for the same vertex. Say for example, if a vertex position is actually  $(10, 10, 10)$ , on one surface it may be claimed as  $(10, 9, 9.5)$ , on the other it may be claimed as  $(9.9, 9, 9.4)$ . We have to assign a single value to the same vertex. We call this vertex unification, and it is done as follows:

1. coalesce vertices on each surface respectively; that is, if  $\text{distance}(\text{vertex1}-\text{vertex2})$  is less than  $\epsilon$ , then merge the two vertices into one.



Line L1 and L2 intercept at vertex A. But the vertex position stored in edge segment L1 is A' and the vertex position stored in edge segment L2 is A''. Because A' and A'' are within an acceptable distance, they are grouped into one position. That is the average of A' and A''.

The new value A will take the place of A' in L1 and A'' in L2

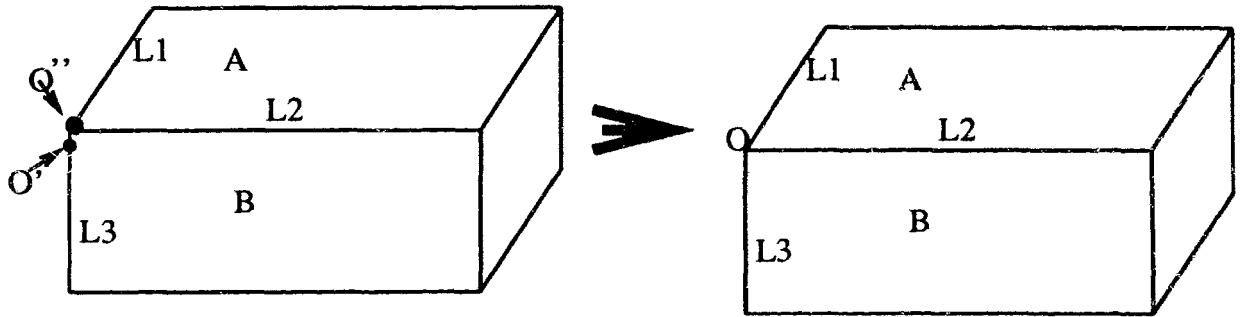
Figure 3.18: Illustration

2. coalesce vertices within a threshold  $\epsilon$  on all surfaces. that is, if  $\text{distance}(\text{vertex1}-\text{vertex2})$  is less than  $\epsilon$ , then merge the two vertices into one.

After the vertex information has been unified in the whole object, the next step is to obtain a closed loop description of the vertices on each face, traversed in a counter-clockwise direction. This implicitly provides the outward normal of the face.

Using the information on which two edges formed the vertex, we can get a counter-clockwise closed loop description as follows:

1. Start randomly with a vertex on a surface. Pick up either one of the edges that formed this vertex to be the next edge. We walk along this edge to the other vertex that forms this edge. Continue this way until we come back to the starting vertex.
2. Next, we make sure that the loop is in counter-clockwise order. This can be achieved as follows:



On surface A, edge L1 and L2 intersect at vertex O', while on surface B edge L2 and L3 intersect at vertex O'. However O' and O'' are so close that they will be grouped into one vertex O.

Figure 3.19: Illustration

- 1) Find the vertex with the biggest j value, say it is vertex  $A_2(i_2, j_2)$ ; in fact, any convex vertex will do.
- 2) Find  $A_2$ 's immediate neighbors. That is, the vertex  $A_1(i_1, j_1)$  that proceeds  $A_2$  and the vertex  $A_3(i_3, j_3)$  that proceeds  $A_2$ .
- 3) If the sign of the determinant calculated in Equation 3.14 is positive, then the loop is already in the counter-clockwise order. Otherwise, invert the loop<sup>1</sup>.

$$S = \begin{vmatrix} j_1 & i_1 & 1 \\ j_2 & i_2 & 1 \\ j_3 & i_3 & 1 \end{vmatrix} \quad (3.14)$$

---

<sup>1</sup>For more on this refer to [16] and [17].

Now, we have extracted the visible surfaces we have in the scene, how many vertices there are on each of these surfaces and the co-ordinates of the vertices. However, since all the vertex processing is done in  $2D$  in the image plane, the  $z$ -component (depth) of the vertices needs to be extracted. Here, we simply refer back to the surface equations to get the  $z$ -component information, more sophisticated schemes (refer to [20]) can minimize the error generated by this conversion.

### 3.3 Summary

In summary, we have proposed and implemented a hybrid edge and region-based approach that extracts boundary descriptions of visible parts of a polyhedral object from a range image. Two main stages are involved in achieving the final description, (i) low-level image processing, and (ii) the geometric analysis. The low-level image processing is used to derive edge segments, and planar regions. These primitives are then used for subsequent geometric “combine and compare” analysis in which potential roof edge segments are derived by intersecting all possible pairs of planar surfaces. These potential edge segments are then validated against the roof edge map to derive the valid roof edge segments. A similar combine and compare approach is used to extract valid vertices on each surface individually. Our approach leads to a relatively simple vertex classification. The final boundary representation is derived by unifying the processing result of each surface. Two main distinguishing features of this approach are (i) combined use of both region and edge-based approaches to extract the geometric primitives, and (ii) that the geometric analysis is carried out on each surface independently instead of the conventional vertex or junction based analyses.

# CHAPTER 4

## Experimental Results and System Evaluation

Three examples are given in this chapter to illustrate the algorithm's implementation. These three examples were chosen because each one represents an interesting feature that the method proposed in this thesis can deal with. All the images used in this thesis are obtained by a 100X white scanner <sup>1</sup>at the PRIP Lab in Michigan State University.

---

<sup>1</sup>For more information about the 100X white scanner see [2]

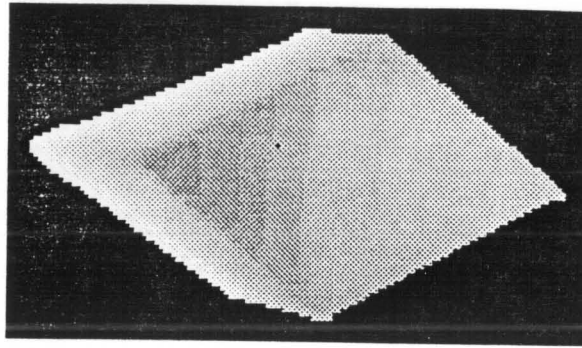


Figure 4.1: "BOX" Image: Original Z Image

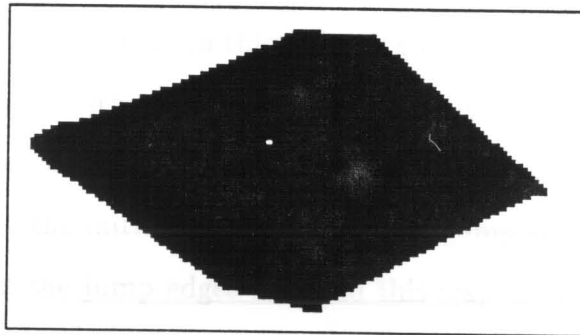


Figure 4.2: "BOX" Image: Flag Image

## 4.1 A Simple Polyhedral Object

Figure 4.1 shows a box (referred to as "box image" later in this thesis) which is of resolution  $81 \times 139$  pixels. In this figure, the range values are shown in gray level. The darker the gray level is, the closer it is to the viewer, and the background in this image is set to be totally black. Figure 4.2 shows its corresponding flag images in which the black area is the region of interest. The white area in this flag image is considered as "background".

The jump edges detected, using the method described in Chapter 4, are shown

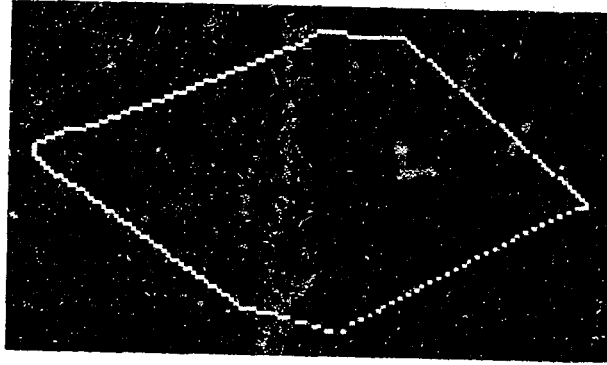


Figure 4.3: "BOX" Image: Jump Edge Map

in Figure 4.3. The process of detecting valid roof edge map is shown in Figure 4.4. The edges shown here have been thinned using the non-maximum suppression edge thinning algorithm. Although they can be further thinned using this algorithm, quite a few broken lines will appear and will also cause more noise in the HT voting. As you can see, in the initial roof edge map, the jump edges are stronger than the real roof edges. If the jump edges detected this way are not removed and the HT is applied here, then the result of the segmentation on the valid edges in the roof edge map will be very bad. Because there exists a quality difference between a roof edge map and a jump edge map, the processing of this two kind of edges should be carried out separately. The separated regions, especially at the boundary, have some noise which is removed by the shrinking method described in Chapter 4.

Region extraction procedures are shown in Figure 4.6. The results of the shrinking process are presented in Figure 4.7. The final result of the processing is shown in Table 4.1 and Figure 4.8.

This "box" image has been chosen as an experimental example because it has slanted surfaces and lines which are typical for polyhedral objects.

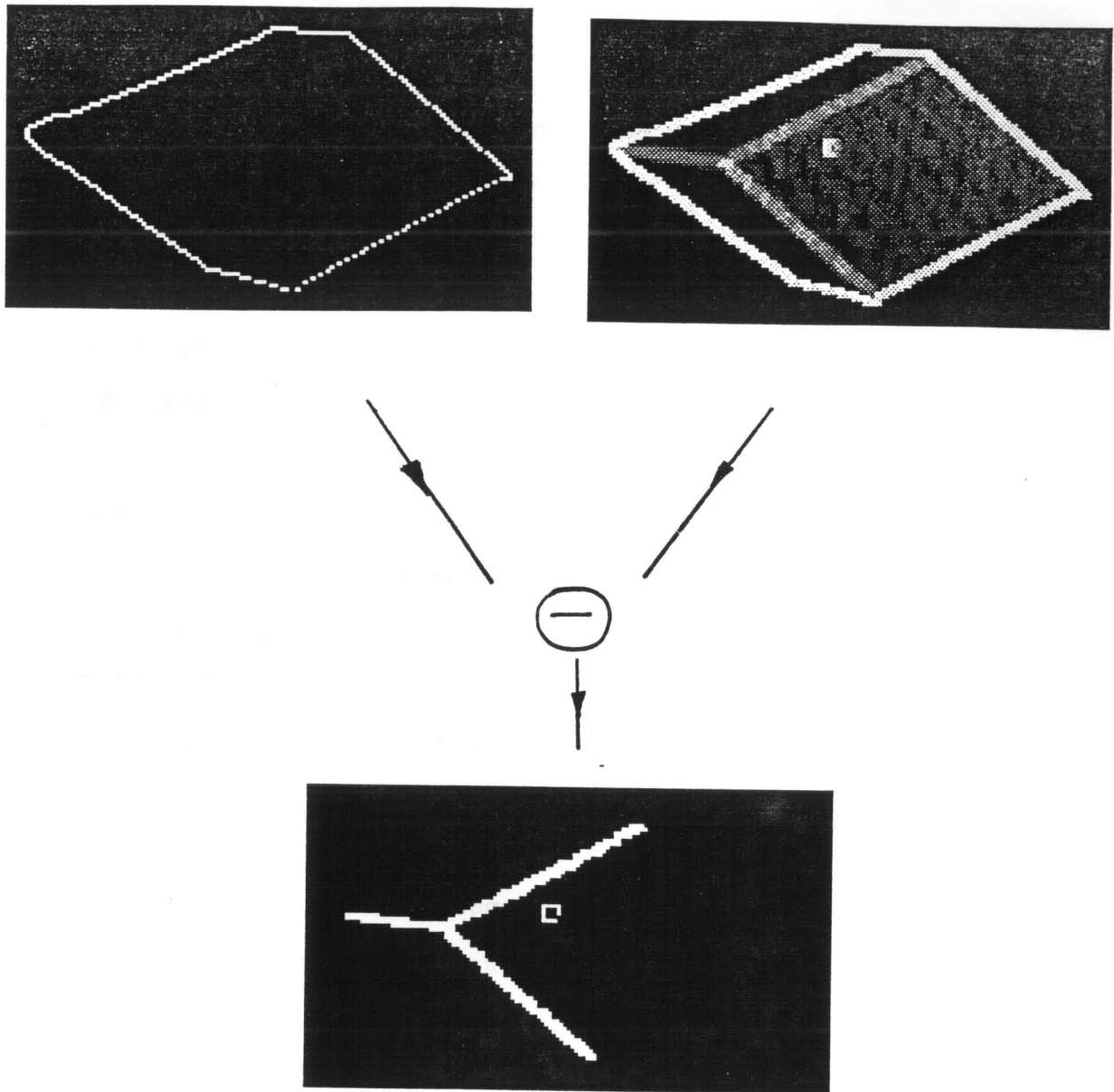


Figure 4.4: The Process of Getting the Roof Edge Map



3 surfaces

on surface 0, there are altogether 4 vertices.

in counter-clockwise order they are:

5 70 260

32 6 267

36 33 117

6 87 180

on surface 1, there are altogether 4 vertices.

in counter-clockwise order they are:

6 87 153

36 33 108

74 76 150

44 130 195

on surface 2, there are altogether 4 vertices.

in counter-clockwise order they are:

71 57 235

74 76 149

36 33 102

32 6 227

Table 4.1: The result of the processing

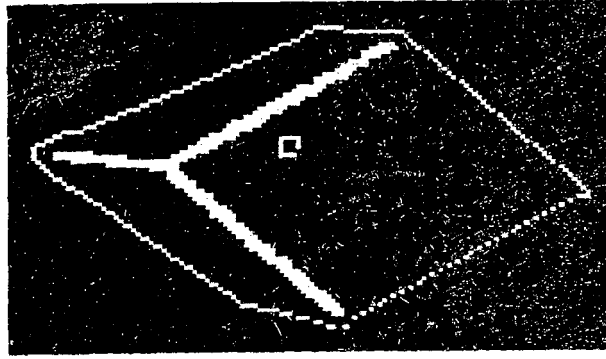


Figure 4.5: “BOX” Image: Combined Edge Map

## 4.2 A Polyhedral object with Vertical and Horizontal Lines

Figure 4.9 shows the image referred to as “Rectangular”. This image was chosen as an example because of its horizontal and almost vertical lines. The horizontal jump edge in this image is relatively difficult to be detected by the original HT technique due to its very short length. The votes for this edge in the HT voting array can be easily surpassed by noise. With the proposed two-step threshold scheme<sup>2</sup>, this short edge can be found. The almost vertical line in the roof edge, shows that to assume the simple  $y = ax + b$  as the line function is not enough. Another parameter is needed. The general edge function should be  $cy = ax + b$ , since it allows the situation where  $c = 0$  while the previous equation does not.

Figure 4.10 shows its corresponding flag image. Its jump edge map is shown in Figure 4.11. The process of getting the valid roof edge map from the initial roof

---

<sup>2</sup>refer to Section 3.2.2 of Chapter 3 for more.

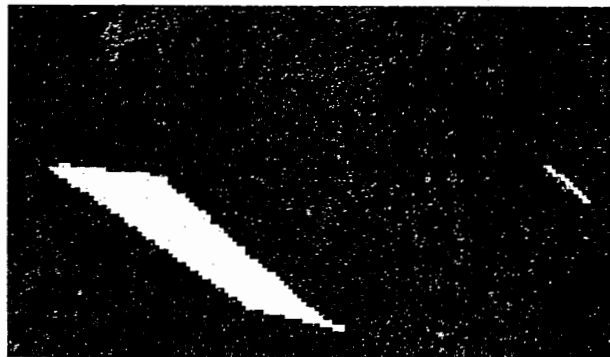
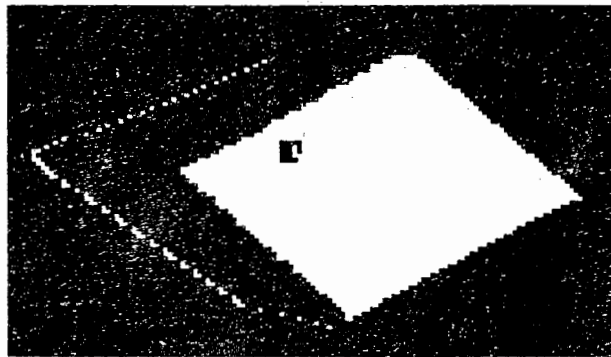
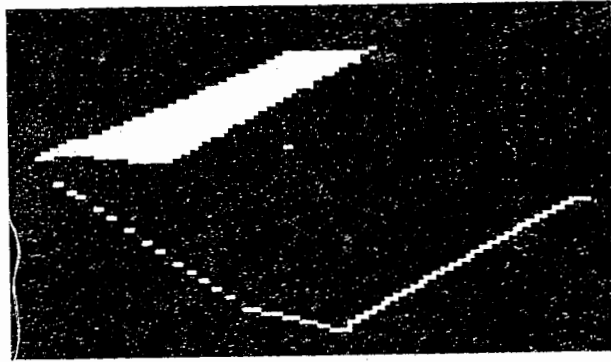


Figure 4.6: "Box" Image: Region Segmentation

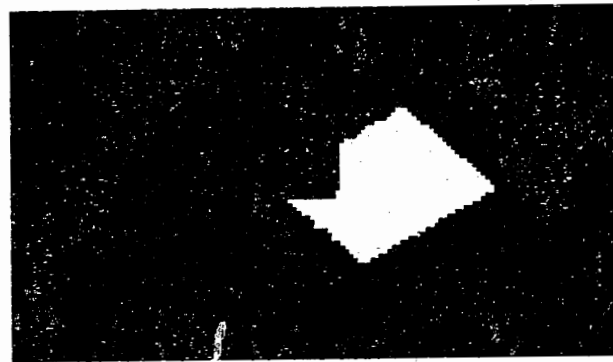


Figure 4.7: "BOX" Image: Region Shrinking

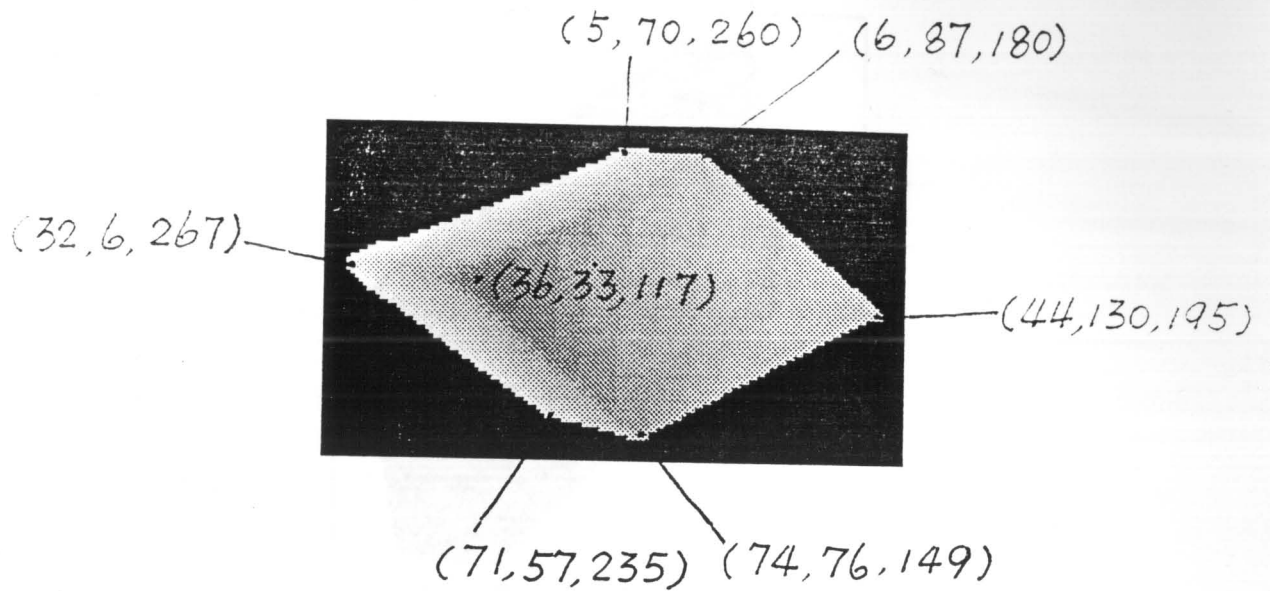


Figure 4.8: Result: Box Image

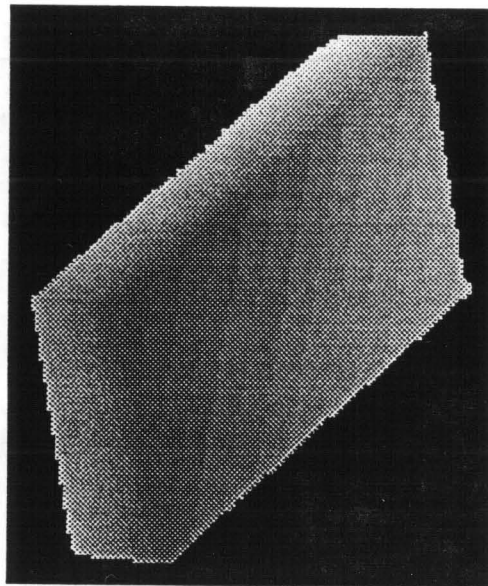


Figure 4.9: "Rectangular" Image: Original Z Image

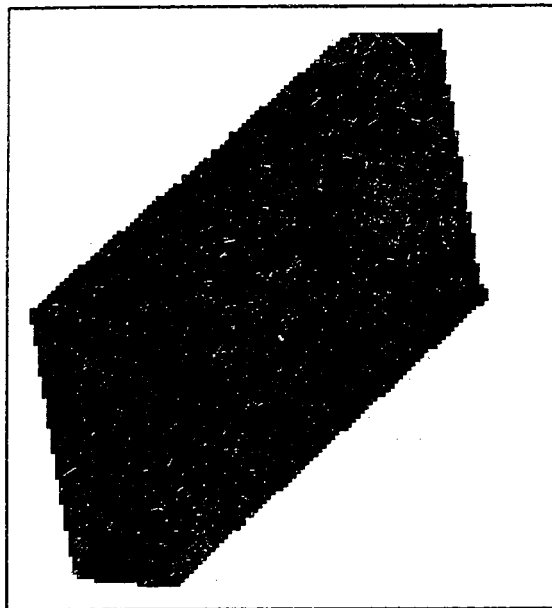


Figure 4.10: "Rectangular" Image: Flag Image

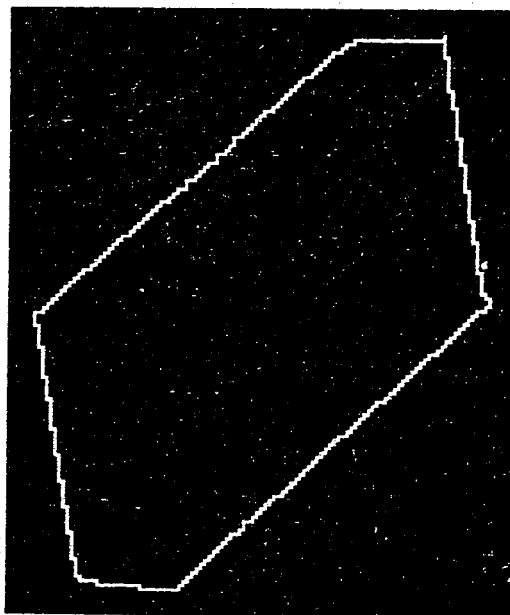


Figure 4.11: "Rectangular" Image: Jump Edge Map

edge is shown in Figure 4.12. Region segmentation and region shrinking are shown in Figure 4.14 and 4.15 respectively.

### 4.3 A Polyhedral Object with a Non-Convex Surface

This image was chosen not only because of its non convex surface, but also due to some of the vertex types it has. Because it has a face with the shape of an “L”, it is called the L image.

The original L image and its corresponding flag image are shown in Figures 4.16 and 4.17 respectively. Figure 4.18 shows the jump edge map and Figure 4.19 shows the initial roof edge, the final roof edge and the process of getting this final roof edge map. Region segmentation and region shrinking experiments are shown in Figures 4.21 and 4.22 respectively.

It is interesting to note the resulting image when the line function of  $AB$  and  $EF$  are grouped together to obtain the potential vertex  $C$ , as shown in Figure 4.23. From previous edge processing, it is known that the start and end points of edge  $AB$  is  $A'$  and  $B'$ . And the start and end points of edge  $EF$  is  $E$  and  $F$ . In order to determine whether or not vertex  $E$  is an existing vertex, the added edge segments of  $B'C$  and  $CE$  are checked to see if they have corresponding pixels in the combined edge map. Because the added segment  $CE$  does not have any corresponding pixels in the combined edge map (see, for example, Figure 4.20), vertex  $C$  is considered an invalid vertex to be removed.

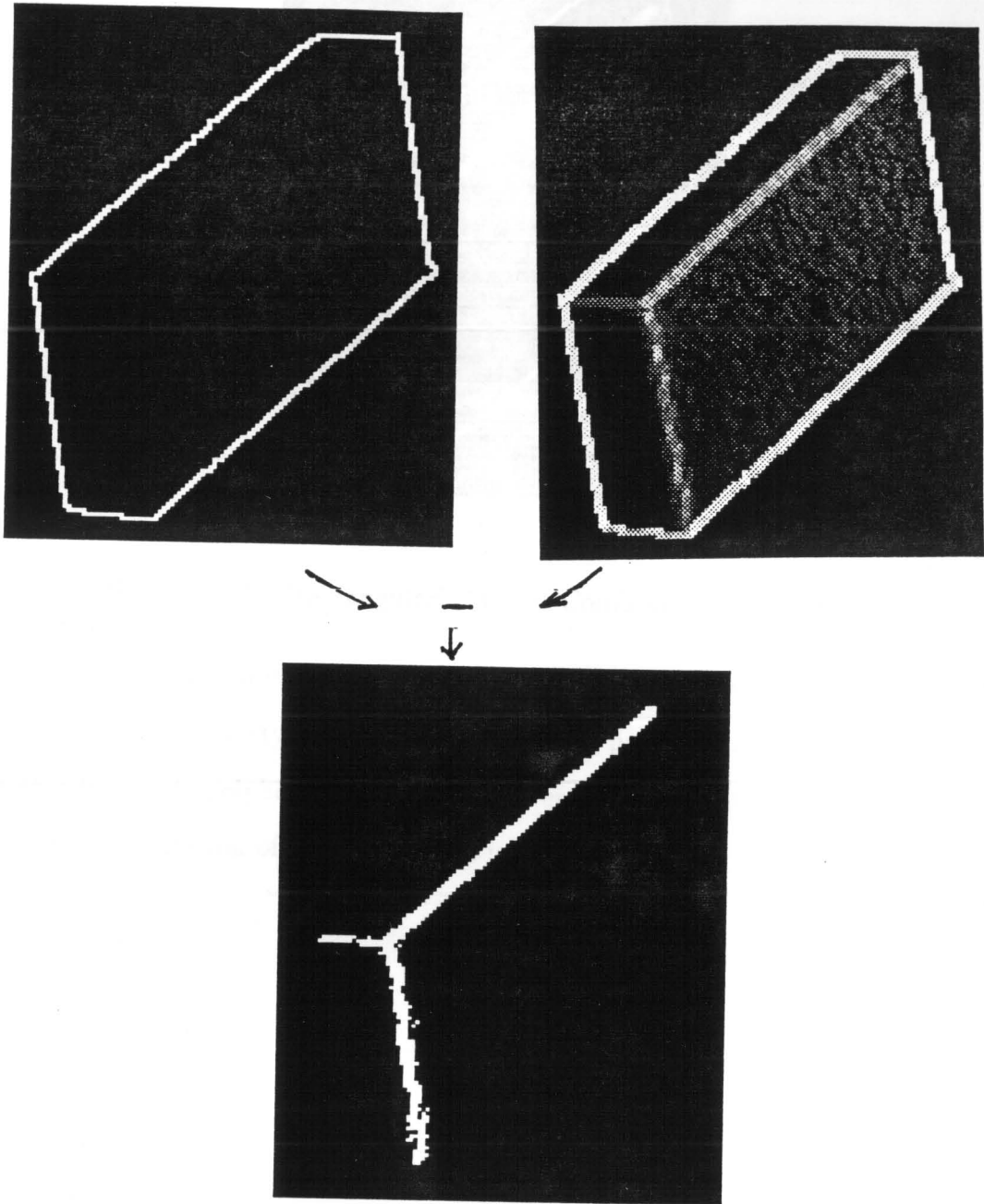


Figure 4.12: The Process of Getting the Roof Edge Map



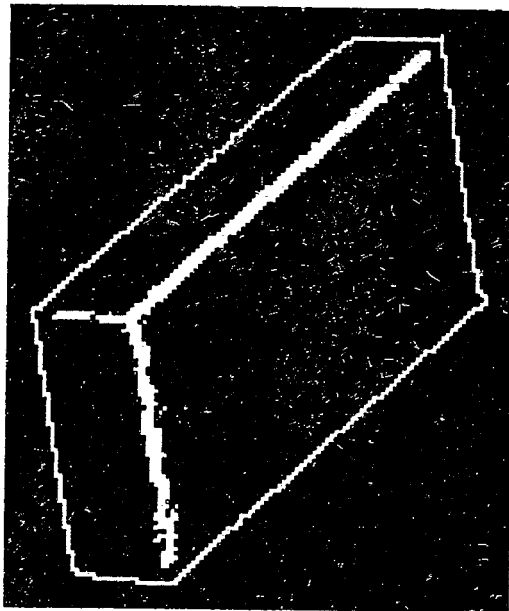


Figure 4.13: “Rectangular” Image: Combined Edge Map

Overall, the above three images illustrate how the method proposed in Chapter 3 works under different line types and different vertex types. Admittedly, our examples are relatively simple polyhedral objects. However, the fact that we can recover the boundary representation of visible surfaces is encouraging.

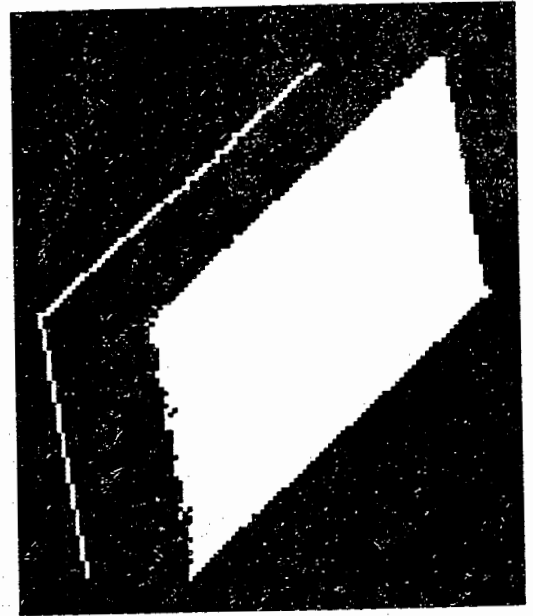


Figure 4.14: "Rectangular" Image: Region Segmentation

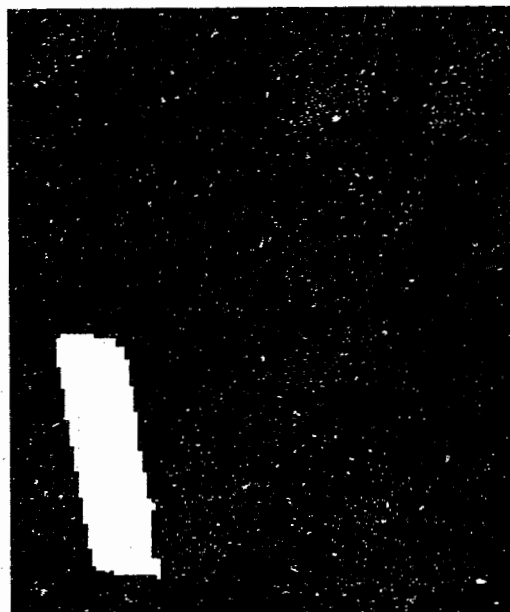
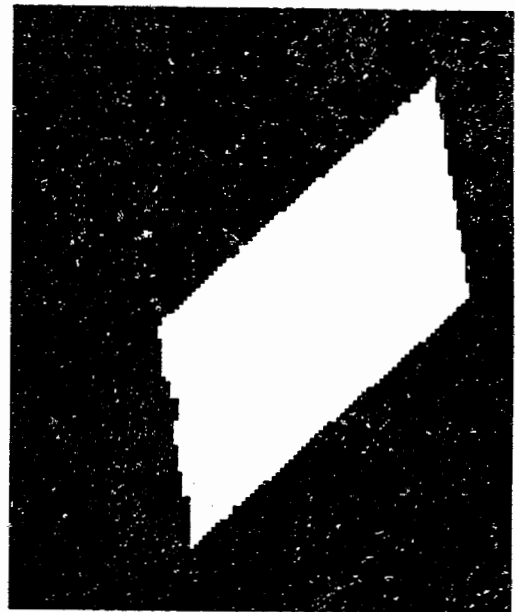


Figure 4.15: "Rectangular" Image :Region Shrinking

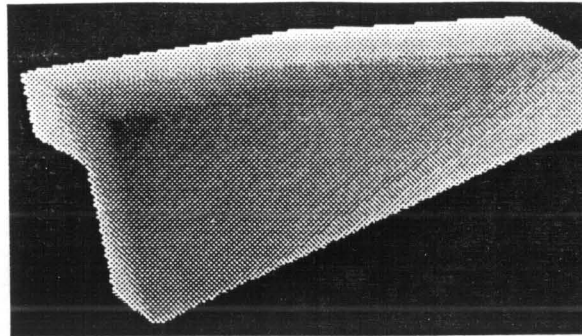


Figure 4.16: "L" Image: Original Z Image

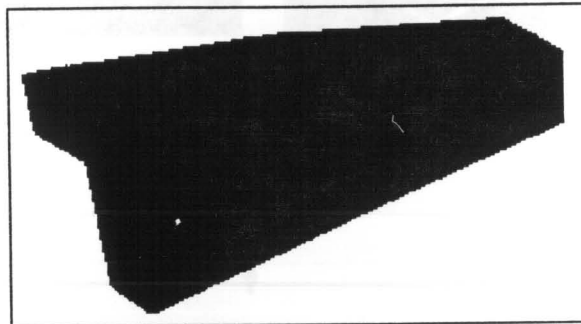


Figure 4.17: "L" Image: Flag Image

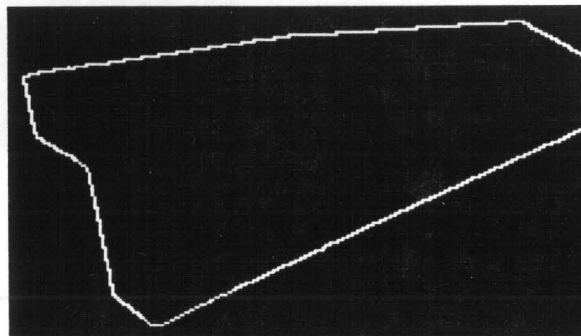


Figure 4.18: "L" Image: Jump Edge Map

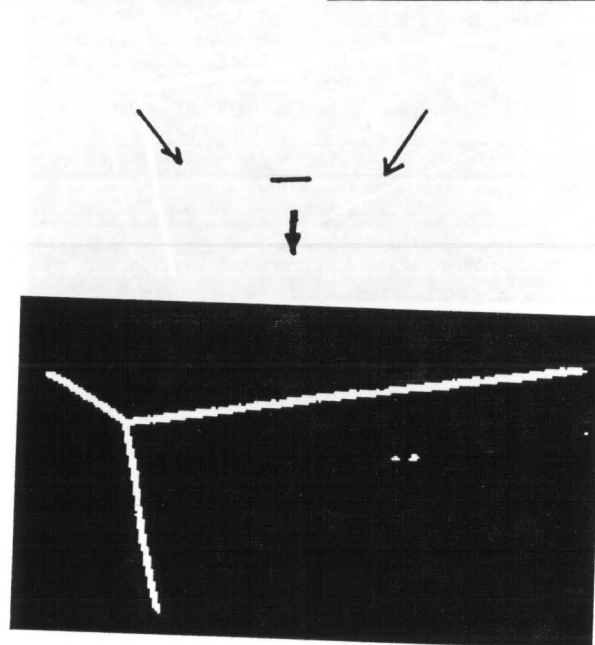
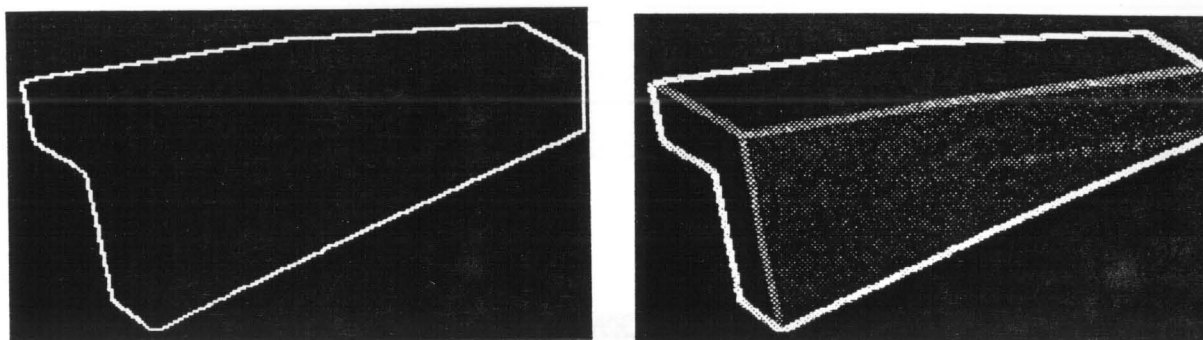


Figure 4.19: "L" Image: The Process of Getting the Roof Edge Map

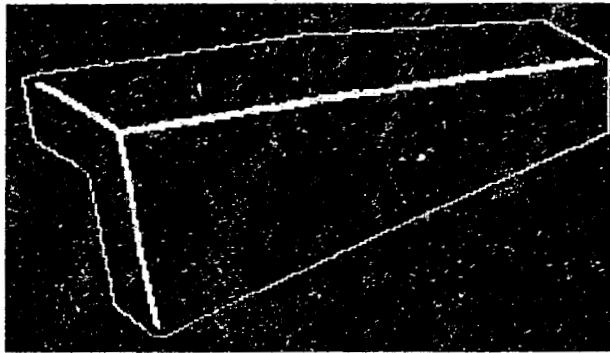


Figure 4.20: "L" Image: Combined Edge Map

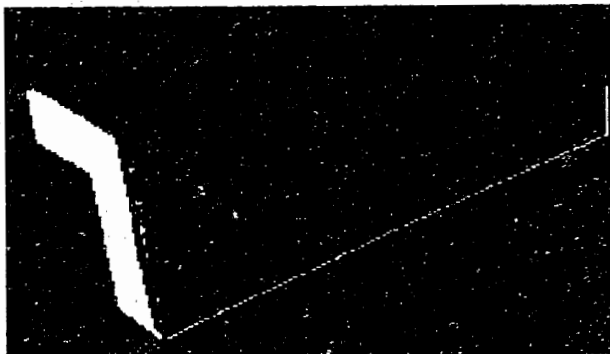
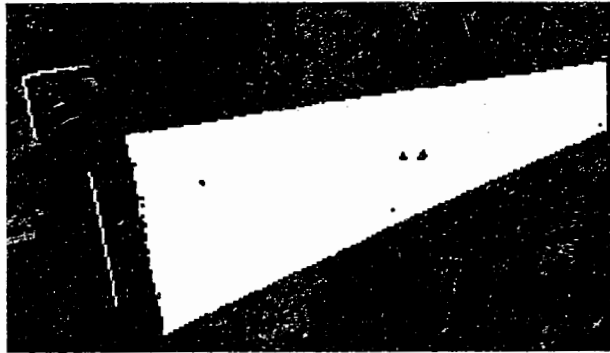
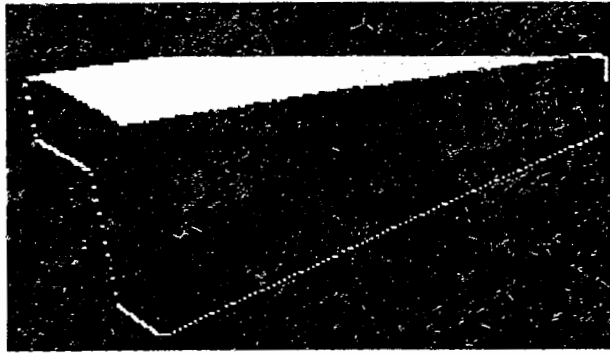


Figure 4.21: "L" Image: Region Segmentation

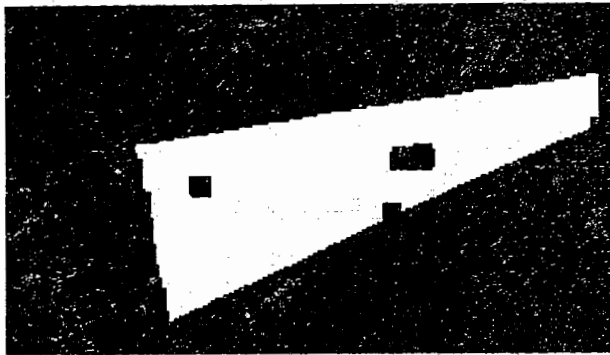
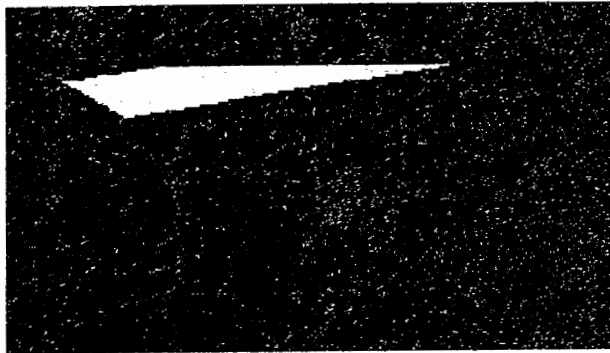
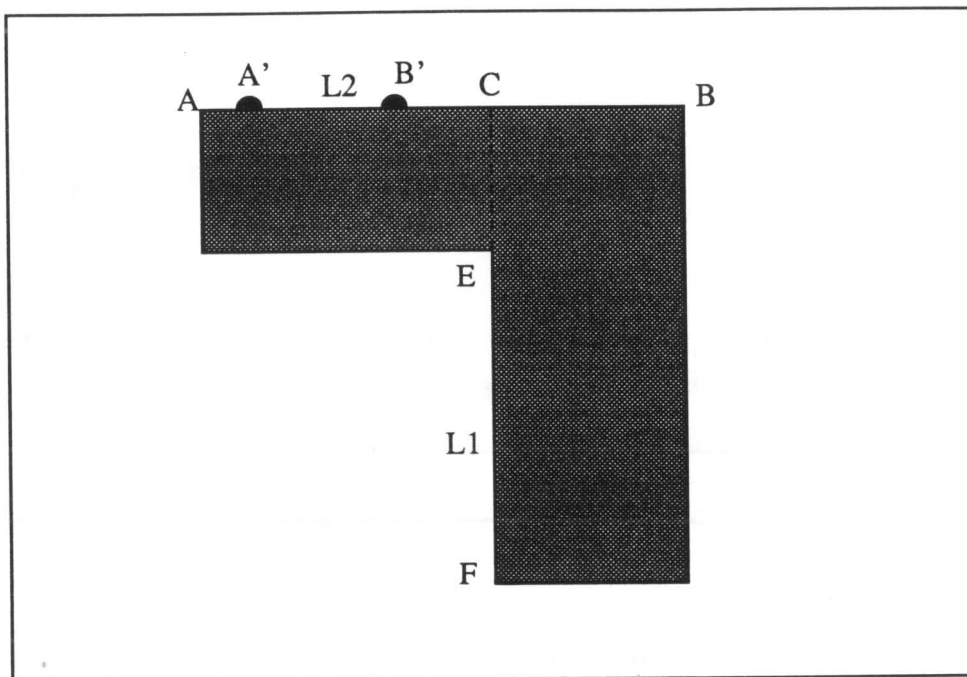


Figure 4.22: "L" Image: Region Shrinking





Line L1 and L2 intersect at C. C is in between vertex A and B.  
Therefore C is not a valid vertex.

Figure 4.23: "L" Image: an interesting example

# CHAPTER 5

## Conclusion

This thesis has presented a unique approach to derive a 3-dimensional description (B-rep type) of visible surfaces of a polyhedral object in a range image. Such descriptions are useful for robotic manipulation tasks such as path planning and grasp planning.

The desired description in a boundary representation is vertices, edges, and faces, their geometry and topological relationships. That is why researchers have primarily used edge-based approaches for deriving the polyhedral descriptions. Almost all previous attempts to recover such 3-dimensional models have used purely edge-based methods at the lower processing level and then tried to combine these edges into a 3-dimensional description. However, there are problems with purely edge-based approaches. For instance, low level edge operators are notorious for generating many false edges and missing many true edges. Although Sugihara's approach of using the junction dictionary may help somewhat, it is not enough.

We believe that implicitly region-based descriptions from the lower processing level contain crucial information needed for building 3-dimensional descriptions of polyhedral objects. Instead of making more assumptions about the robot working environment and further improving the edge-based techniques, this thesis explored the use of region-based approaches in conjunction with edge-based approaches to derive 3-dimensional polyhedral descriptions.

This thesis shows a way of combining the information from the edge-based methods and the region-based methods to obtain 3-dimensional boundary descriptions of polyhedral objects. Its main contribution lies in that it uses information from both edge-based and region-based methods at lower level processing to facilitate the further geometric analysis of vertices, edges and faces of the polyhedral object. While most other researchers mainly focus on improving the edge-based method, such as the HT, to get a more reliable surface information, we use the region-based method to get the surface information directly from the raw data. We combine all the surface equations to get possible edges. We then compare the possible edges detected above with the original roof edge map to remove the false roof edges. We call this “combine and compare” approach and use it in particular to get the roof edge segments.

This “combine and compare” method is also applied to derive the location of all possible vertices. In order to distinguish valid and invalid vertices, this thesis presents a new way of defining the valid vertices.

This thesis also emphasizes the separation of processing of different regions and types of edges. The point that each surface region, and each edge type should be processed separately is stressed. By doing so, geometric analysis is greatly simplified.

As it exists, the proposed system has some limitations. It has been tested on simple objects where all surfaces are completely visible or invisible. Although in theory, we believe, the proposed system should be able to give descriptions of all visible parts of an object, no test has been done on objects with partially visible surfaces or objects with holes. We will need to extend our boundary representation to include partially visible faces. Also, in order to get a better final description, this approach requires that low-level processing provide good quality results. This is especially true for segmentation. A good low-level segmentation is crucial for geometric analysis later on. The segmentation implementation carried out in this thesis is relatively simple and straightforward. More sophisticated approaches should be used in order to improve the quality of segmentation. At present, human interaction is needed to get a good quality segmentation.

There are many major open issues in this thesis research that need further exploration. First, low level processing issues. The edge detection and surface segmentation techniques used in this thesis are relatively “naive”. More robust techniques could be used here. Next, we have used a very simple type of B-rep model for polyhedral surfaces – simply a list of each (oriented) face of the polyhedron. This should be extended to more general polyhedral surfaces with holes. Another major issue is that of extracting a limited class of curved surfaces, e.g., B-spline based surfaces. Another issue is to deal with edges that are missed by lower level image processing as discussed in Chapter 3. We could incorporate Sugihara’s junction dictionary for polyhedral domains. However, the method presented in this thesis should also be able to deal with some missing edges in the processing.

Two rather simple examples are given in Figure 5.1 and Figure 5.2 respectively.

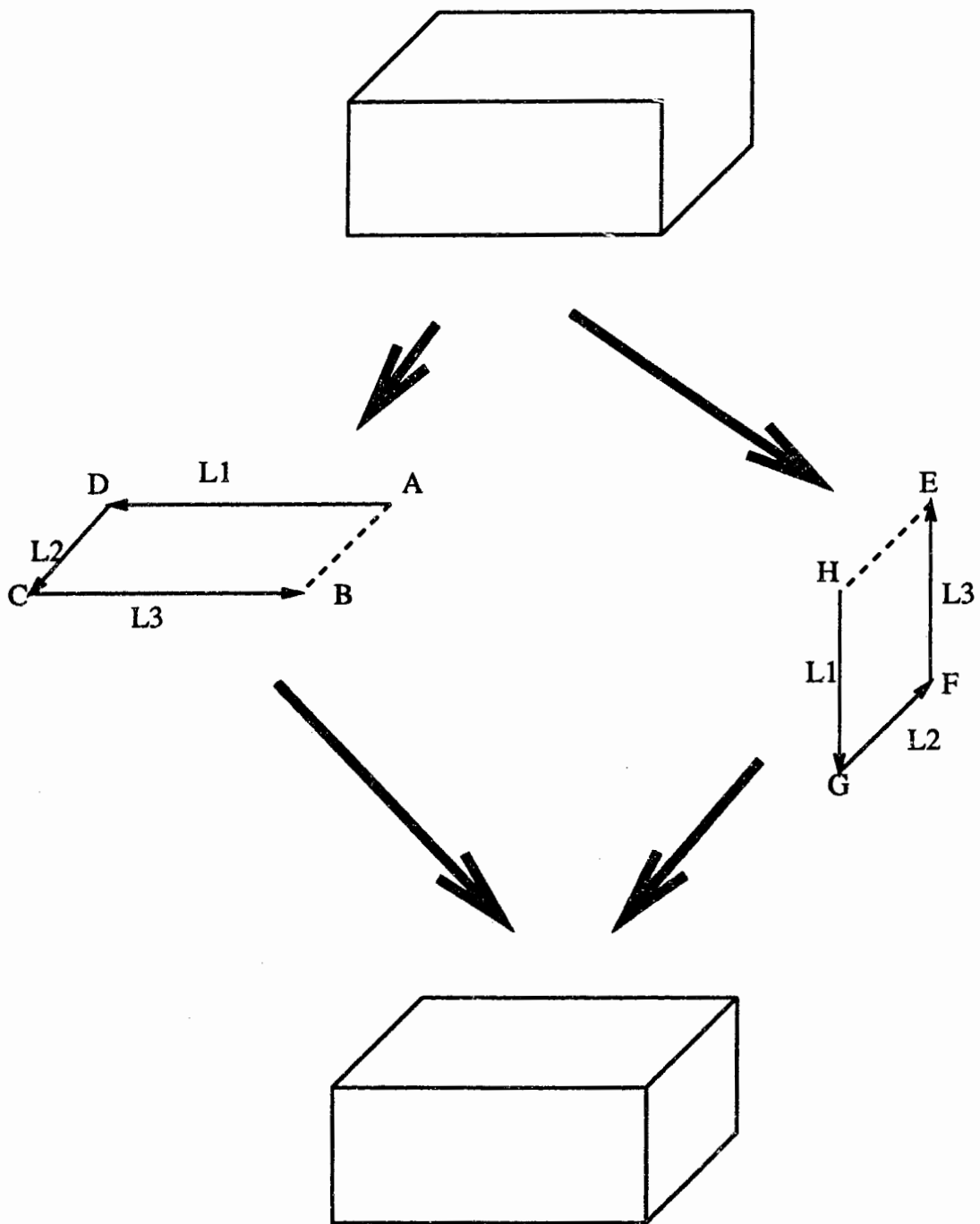


Figure 5.1: Example: One surface has no more than one missing edge

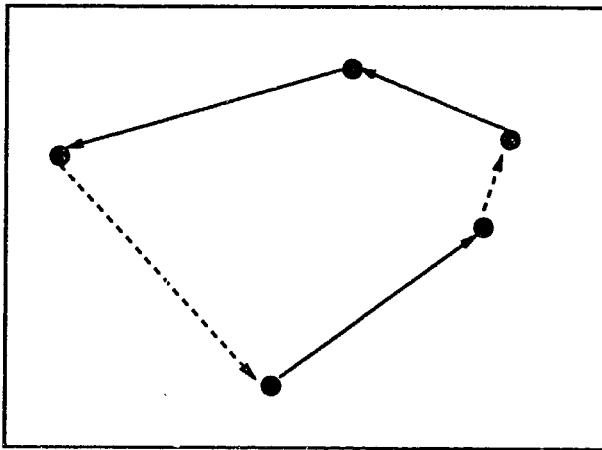
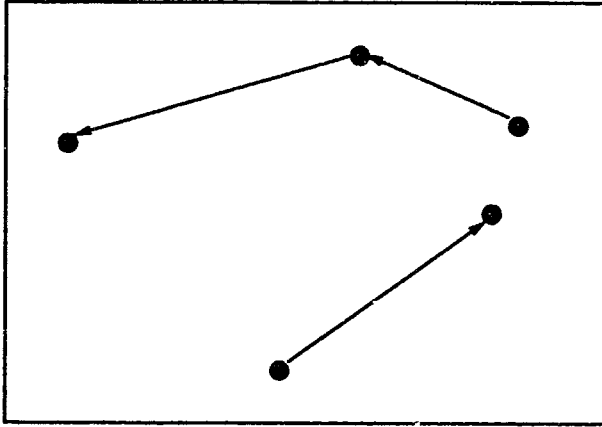


Figure 5.2: Example: One surface has more than one missing edge

The first example shows the situation where only one edge is missing on one surface, the recovery of this edge is almost guaranteed. This is done by checking if a vertex is a closed vertex, which means checking to see if there is an edge coming into the vertex and another edge going out from it. If both of these edges exist, then it is a closed vertex. If not, this unclosed vertex should be one of the vertices that belong to the missing edge. If each missing edge is on a different surface, it can be retrieved individually on its respective surface, as described above. The second example shows that this recovery process, at least in some cases, may also be applied to situations where more than one missing edge exist on the same surface. These are just examples that illustrate a need for developing a more comprehensive methodology to use our approach to recover missing edges.

# REFERENCES

- [1] D.H. Ballard and C.M. Brown, **Computer Vision**, *Englewood Cliffs, N.J.: Prentice-Hall, c1982*
- [2] T.E. Beeler, **Producing Space Shuttle Tiles with a 3-D Non-Contact Measurement System**, *Three-Dimensional Machine Vision*, pp. 513–541.
- [3] P. Besl, **Surfaces in Range Image understanding**, *Springer-Verlag New York Inc. 1988*
- [4] J. Canny, Master Thesis, MIT-TR720
- [5] D. Marr, **Vision : a computational investigation into the human representation and processing of visual information**, *San Francisco : W.H. Freeman, 1982*
- [6] Davies and Yarwood, **Engineering Drawing and Computer Graphics**, *Wokingham, Berkshire, England: Van Nostrand Reinhold(UK) Co., 1986.*
- [7] R.O. Duda, D. Nitzan and P. Barrett, **Use of Range and Reflectance Data to Find Planar Surface Regions**, *IEEE Transaction on Pattern Analysis and Machine Intelligence* , PAMI-1, 3, pp. 259–271, July 1979
- [8] M.C. Fairhurt, **Computer Vision for Robotic System: An Introduction**, *New York; London: Prentice Hall, 1988*
- [9] O.D. Faugeras, and M. Hebert, **A 3-D Recognition and Positioning Algorithm Using Geometrical Matching Between Primitive Surfaces**, *Proc. Eighth International Joint Conference on Artificial Intelligence*, August, 1983, pp.1108–1112
- [10] J.D. Foley, A. van Dam, S.K. Feiner and J.F. Hughes, **Computer Graphics: Principles and Practice**, *2nd Edition, Massachusetts, California, New York: Addison-Wesley Publishing Company, 1990*



- [11] R.C. Gonzalez, **Digital Image Processing**, *Massachusetts: Addison-Wesley Publishing Company, Advanced Book Program*, 1977
- [12] R.C. Gonzalez and P. Wintz, **Digital Image Processing**, *2nd edition, Massachusetts: Addison-Wesley Publishing Company*, 1987
- [13] K.K. Gupta, **Fast collision avoidance for manipulator arms: a sequential search strategy**, *IEEE Transactions on Robotics and Automation*, Vol. 6, No. 5, 1990
- [14] K.K. Gupta and Z. Guo, **Toward Practical Motion Planners: Experiments with A Sequential Search Strategy**, '91 ICAR Fifth Inter. Conf. on Advanced Robotics, Vol. 2, pp 1006 - 1011, June 1991
- [15] O.D. Faugeras and M. Hebert, **Segmenting Range Data into Planar and Quadratic Patches**, *CVPR, Washington*, June, 1983
- [16] M.A. Hill, Jr. and J.B. Linker, **Brief Course in Analytics**, *Holt, Rinehart and Winston: New York, Chicago, San Francisco, Toronto and London. 3rd Edition, 1960*
- [17] P. J. Kelly and E. G. Straus, **Elements of Analytic Geometry**, *Scott, Foresman And Company: Glenview, Illinois. 1970*
- [18] Donald Hearn and M. Pauline Baker, **Computer Graphics**, *Englewood Cliffs, New Jersey: Prentice-Hall*, 1986
- [19] M. Hebert and J. Ponce, **A New Method for Segmenting 3-D Scenes**, *Proc. of the Sixth ICPR*, pp836-838, Munich, October, 1982
- [20] M. Herman, **Generating Detailed Scene Descriptions from Range Images**, *85 IEEE International Conference on Robotics & Automation*, pp 426 - 431
- [21] B.K.P. Horn, **Robotic Vision**, *Cambridge Massachusetts; London, England: The MIT Press*, 1986
- [22] R.O. Duda and P.E. Hart, **Use of the Hough Transformation To Detect Lines and Curves in Pictures**, *Communications of the ACM* 15, 1(January 1972), pp11-15
- [23] S.L. Horowitz and T. Pavlidis, **Picture Segmentation by a Tree Traversal Algorithm**, *Journal of the Association for computing Machinery*, Vol 23, No. 2, pp368-388, April 1976

- [24] A. Klinger, **Data Structures and Pattern Recognition**, *1st International Joint Conference in Pattern recognition, Washington D.C., Oct. 1979* pp 497-498
- [25] R.H. Laprade, **Split-and-Merge Segmentation of Aerial Photographs**, *CVGIP Vol 44, pp77-86, 1988*
- [26] K.V. Mardia and T.J. Hainsworth, **A Spatial Thresholding Method for Image Segmentation**, *IEEE Transacation - PAMI vol 10, pp919-927, 1988*
- [27] M. E. Mortenson **Geometric Modeling**, *New York, Chichester, Brisbane, Toronto, Singapore: John Wiley & Sons, 1985*
- [28] M. Oshima and Y. Shirai, **A Scene Description Method Using Three-Dimensional Information**, *Pattern Recognition 11 (1979)*, pp9 - 17
- [29] T. Lozano-Perez, **A simple motion planning algorithm for general robot manipulators**, *IEEE Journal of Robotics and Automation. Vol.3 No. 3, June 1987*
- [30] J. Ponce and M. Brady, **Towards a Surface Primal Sketch**, *Three-Dimensional Machine Vision (T. Kanade, Ed.) Kluwer Academic Publishers, 1987, pp 195 - 240*
- [31] T.V. Robertson, **Extraction and Classification of Objects in Multispectral Images**, *Machine Processing of Remotely Sensed data, IEEE 73 Cho 837-2GE, Purdue University, Indiana, October 16-18,1973, P3B-27 to 3B-34*
- [32] K. Sugihara, **Range-Data Analysis Guided by a Junction Dictionary**, *Artificial Intelligence Vol 12, pp41-69, May, 1979*
- [33] K. Sugihara, **Use of Vertex-Type Knowledge for Range Data Analysis**, *Three-Dimensional Machine Vision (Takeo Kanade, Ed), Boston: Kluwer Academic Publishers, pp267-298, 1987*
- [34] R.R.Tito, **The Shape of SAR Histograms**, *CVGIP, Vol 43, 1988, pp 281 - 293*
- [35] R.Y. Wang and K. Hayrepetian, **Image Processing with Intensity and Range Data**, *Proceedings of Pattern Recognition and Image Processing Conference, Las Vegas, Nevada, June 14-17, pp. 518-520*
- [36] N. Yokoya and M.D. Levine, **Range Image Segmentation Based on differential Geometry: A Hybrid Approach**, *IEEE Transcation on Pattern Analysis and Machine Intelligence , Vol. II, No. 6, pp. 643-649, June 1989*