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.

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.

Robust Estimation

for

Range Image Segmentation and Fitting

Xinming Yu

A Thesis

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy at

Concordia University

Montréal, Québec, Canada

February 1993

National Library of Canada

Acquisitions and Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada

Direction des acquisitions et des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

ISBN 0-315-84686-0

Abstract

Robust Estimation for Range Image Segmentation and Fitting

Xunning Yu, Ph.D.

Concordia University, 1993

In the dissertation a new *robust estimation* technique for range image segmentation and fitting has been developed. The performance of the algorithm has been considerably improved by incorporating the *genetic algorithm*.

The new robust estimation method *randomly* samples range image points and solves equations determined by these points for parameters of selected primitive type. From K samples we measure *RESidual Consensus* (RESC) to choose one set of sample points which determines an equation best fitting the largest homogeneous surface patch in the current processing region. The residual consensus is measured by a *compressed histogram* method which can be used at various noise levels. After obtaining surface parameters of the best fitting and the residuals of each point in the current processing region, a *boundary list* searching method is used to extract this surface patch out of the processing region and to avoid further computation. Since the RESC method can tolerate more than 80% of outliers, it is a *substantial* improvement over the least median squares method. The method segments range image into planar and quadratic surfaces, and works very well even in smoothly connected curve regions.

A genetic algorithm is used to *accelerate* the random search. A large number of *offline average* performance experiments on GA are carried out to investigate

different types of GAs and the influence of *control parameters*. A *steady state* GA works better than a *generational replacement* GA.

The algorithms have been validated on the large set of synthetic and real range images.

Acknowledgments

The research work described in this dissertation is made possible by the continuous support and guidance of my supervisors, Professors T. D. Bui and A. Krzyżak. Through several years of my graduate studies at Concordia University, Professor Bui has been constantly supportive during the hard times and encouraged me to pursue the high level research work. Many discussions with Professor Bui benefited me in developing methodology of scientific research, improved my ability for problem analysis and solving. The two courses, *Pattern Recognition and Image Processing* and *Robot Vision*, given by Prof. A. Krzyżak give me solid foundations in the area. The projects in the courses triggered my interest in research in computer vision and image processing. I deeply appreciate all the help given to me by my supervisors. I would also like to thank Dr. T. Kasvand for his very helpful guidance and discussions. I also thank Professor Suen for his initial guidance and admitting me to the Ph.D. program. I am grateful to Dr. E. Regener for help in revising part of the dissertation.

Many research associates and graduate students in the Center for Pattern Recognition and Machine Intelligence and in the Laboratory for Scientific Computing of Concordia University helped me a lot. Dr. Lei Xu had given me many useful ideas and broadened my knowledge during his visit. I highly value his many comments and practical advises. I thank Ms. Peiying Zhu for many helpful discussions and comments. I would like to thank Mr. Xianjun Wang and Deming Li for their very helpful discussions and support in some mathematical derivations and finding program libraries. I would also thank Mr. Marc Pawlowsky for the helpful discussion about genetic algorithms. I also thank Mr. Jiawei Guo, Dr. Y. Y. Tang, Professor Quanlin Gu, Mr. Zixi Li, Mr. Pankaj Kamthan, Mr. John Marshall, Mr. Bill Wong and others.

I thank the Photonics and Sensors Group in Division of Electrical Engineering at the National Research Council of Canada for providing range images. Without these images, the research work would not be possible. Also I would like to thank Pattern Recognition and Image Processing Laboratory of Michigan State University for providing real and synthetic range images in the public domain.

I highly appreciate the financial support from my supervisors, Professors T. D. Bui and A. Krzyżak, and from the Centre de Recherche Informatique de Montreal Inc.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> Computer vision is the science that develops the theoretical and algorithmic basis by which useful information about the world can be automatically extracted and analyzed from an observed *image, image set*, or *image sequence* from computations made by special purpose or general purpose computers. Such information can be related to the recognition of a generic object, the three-dimensional description of an unknown object, the position and orientation of the observed object, or the measurement of any spatial property of an object, such as the distance between two of its distinguished points or the diameter of a circular section. Applications of the technology range from vision guided robot assembly to inspection tasks involving mensuration, verification that all parts are present, or determination that surfaces have no defects.
>
> Robert M. Haralick and Linda G. Shapiro [illegible]

In the past 35 years, significant advances have been made in the field of computer vision, but machines still fall *far short* of humans and animals in their visual performance [1]. Many scientists and engineers devote their great effort to solve this difficult problem.

In this dissertation, efforts have been made in robust estimation, genetic algorithms, range image segmentation and fitting, quadratic surface invariants and pose determination. We have been making some progress in these areas.

An object can normally be described by a set of geometric primitives, which can be in the form of the first-order (planar surface), the second-order (quadratic surfaces) or higher order surfaces. Robust estimation is a proper way to extract primitives from noisy data. A genetic algorithm can be used to accelerate such robust estimation technique. Range image then can be segmented into geometric primitives. The poses of quadratic surfaces then can be determined for recognition.

We will survey various methods in these areas and propose new ones to solve the problems in different ways. Each method has its advantages and disadvantages. We analyze the proposed methods and specify their appropriate application domains.

Section 1.1 summaries major contributions of the dissertation and section 1.2 describes organization of the dissertation.

1.1 Major Contributions of the Dissertation

This dissertation consists of two parts: (1) a new *robust estimation* technique for range image segmentation and fitting; (2) *genetic algorithm* (GA) incorporated into the new method to accelerate the search. Major contributions of each part are as follows.

1.1.1 Robust estimation technique (RESC)

Robust estimation technique (RESC) is presented in Chapters 3, 4, 6 and 7. RESC stands for *RESidual Consensus* — a technique for optimization based on residual analyses. The major contents of this part have been published in the *Proceedings of IEEE 1992 Computer Vision and Pattern Recognition* [86], *Proceedings of the SPIE Advances in Intelligent Robotic Systems, Sensor Fusion IV: Control Paradigms and Data Structures* [83] and *Proceedings of the Canadian Conference on Electrical and Computer Engineering* [84]. It has also been submitted to *The IEEE Transactions on Pattern Analysis and Machine Intelligence* [87]. The major contributions in this part are:

1. A new robust estimation technique (RESC) with the following features:

   - High robustness to outliers: the breakdown point of the estimator can be as high as 80% for normal noise levels and 94% in noise free environment. It is much better than LMS method which has a breakdown point of 50%.

   - Better performance in second order primitive estimation: the RESC has been shown to be the best compared with LMS (Least Median Squares method) and LS (Least Squares method) for solving implicit equation under Gaussian noise. LS does not minimize the proper geometric residuals of the implicit equations of the second order primitives. LMS uses a weak criterion for the optimization where the outliers are less than 50%.

   - Ability to handle various noise levels: the compressed histogram method in RESC can handle estimation under different noise levels. This is very important because different sensors have different error levels. Even for the same sensor, the image may have different noise levels for different regions.

   - Easy separation of inliers and outliers: a cutting point can be determined from the compressed histogram by analysing residuals in the histogram

The cutting point separates inliers from outliers.

- Efficiency: histogram technique has time complexity of $\mathcal{O}(n)$ which is better than LMS's sorting $\mathcal{O}(n \log_2 n)$, where n is the number of points in the processing region.

2. Successful application of RESC algorithm in the area of two-dimensional range image profile and three-dimensional range image analysis. RESC is used to extract geometric primitives (planar and quadratic surface patches) from range images. It is also used to extract line and conic segments from range image profiles in two dimensional cases.

3. A two-stage segmentation strategy. A preliminary segmentation is applied to the raw image to detect jump-edges. Smaller regions segmented by jump-edges are more easy to process by the RESC method. Primitive extractions by RESC method is performed in each region. After a primitive is determined, a segmentation algorithm is used to segment the primitive from the region.

4. A segmentation algorithm which can tolerate outliers and works efficiently. The algorithm uses two lists which store boundaries of the processing region. Four-neighbor connectivity is used. The segmentation algorithm will select a largest region in which all residuals are within the estimated noise level.

5. An erosion algorithm to eliminate small region which may be a hole in a continuous region. The small region occur either at the boundary or inside a region where the residual is greater than the fitting threshold. They should be eliminated

6. Demonstration of several complete results of segmentation and fitting of real range images. The results are very good. We can hardly see the difference between the original and the reconstruction, except that the reconstructed range image does not have the noise effect.

7. Development of a complete system for the range image segmentation and fitting, as well as the realistic rendering graphics software for range image visualization with different light sources and shading.

1.1.2 Genetic algorithm (GA)

Genetic algorithm (GA) is presented in Chapter 5. Major contents of this part will be published in the *Proceedings of the 8th Scandinavian Conference on Image Analysis* [88]. The major contributions in this part are.

1. Incorporation of genetic algorithm (GA) into the RESC method to accelerate search. In comparative experiments, we have found that GA had much better performance than a pure random search with the same number of function evaluations.

2. Analyzing the binary gene representation for point indices. The crossover operator will break the string of binary genes which represent an integer, giving a very large equivalent mutation rate. Therefore, the point indices are used directly as genes instead of binary encoding the integers.

3. Testing extensively the genetic algorithm parameter settings to obtain optimal performance of GA. Two different GAs have been tested and compared. The optimal parameter settings were found to be different from what has been suggested in literature. The results are analyzed.

1.2 Organization of the Dissertation

A brief introduction to the chapter is at the beginning of each chapter to give readers an idea of what the chapter is about. A summary or conclusion of each chapter is at the end.

A brief introduction to each chapter is as follows.

In Chapter 2, we extensively review the most commonly used methods for *segmentation* and *surface fitting* for range images. The segmentation and fitting methods are divided into two categories: (1) first segmentation then fitting, (2) first fitting then segmentation. Each category has its advantages and disadvantages. Segmentation and fitting are fundamental processes in the computer vision because they are normally the first steps of the whole system. The segmentation and fitting method proposed in this dissertation falls into the second category.

In Chapter 3, the basic estimation model and analysis are defined and explained. Estimation analysis is the method for finding the best estimation of parameters of a given model from the data set. Various criteria can be used in different applications. In recent years, robust estimation has been greatly emphasised. A robust estimation is an estimation which can still correctly estimate parameters of a given model when there exits outliers. A formal definition of outlier and breakdown point is given in this chapter. The breakdown point is a quantitative measure of the robustness of an estimator. We also analysed the problem with the least squares method for estimation of the second order primitives. For the second order primitive model, it is impossible to express the model in an explicit linear equation in order to use the linear least squares method to minimize the geometric residuals. The normally used equation form is an explicit equation. The least squares method minimizes only the difference between two sides of the equation, called algebraic distance. Furthermore, the least squares method cannot tolerate outliers.

In Chapter 4, we propose a new high breakdown point robust estimator (RESC). The principle of the random sample methods is introduced. It has the advantage of outlier insensitivity. The proposed method is based on the random sample principle and emphasizes *REsidual Consensus* (RESC). The method can be used to segment range image into surface patches and to obtain their equations at the same time. We use *histogram* to analyze *distributions* of *residuals* — the z direction distances between raw data points and the hypothesized surface patches. We introduced a *compressed histogram* method which works on different noise levels. *Histogram power* is calculated and used as our object function of the optimization process. We can always segment the *largest best matched* surface patches from others, even in the case of *smoothly* connected quadratic surface patches with normally distributed *sensor errors*. The most important improvement of RESC algorithm over LMS method is that RESC can tolerate more than 80% outliers. This is demonstrated in the experiments in Chapter 7. For the time complexity of the algorithm, RESC is also an improvement over LMS, because RESC uses histogram analysis which has the time complexity of $\mathcal{O}(n)$, whereas the sorting part of the LMS takes $\mathcal{O}(n \log_2 n)$.

In Chapter 5, we incorporate genetic algorithm (GA) into RESC method. Genetic algorithms are a class of optimization techniques that gain their name from a similarity to certain processes that occur at the interactions of biological genes. Various GA operators and algotithms are introduced. GA accelerates the searching process of RESC method. Integer genes are used instead of binary *ones*. We analyzed the situation where binary genes are used. Crossover operator breaks, with very high probability, the binary string which represents an integer. Such break is equivalent to a mutation operation. An equivalent mutation rate for a n point crossover operator is calculated. Since our gene is an integer and its value is varied in a large range, the GA parameter settings is different from those suggested in the literature where all analyses and experiments are based on binary genes. Extensive experiments are performed in order to determine the parameter setting of GA which is essential for GA working properly.

In Chapter 6, details of the segmentation algorithm are explained. A two-stage segmentation strategy is used. In the first stage, a simple jump edge detector is used to segment preliminarily the whole image into several regions separated by these jump edges. This can effectively reduce the amount of computations of the RESC algorithm. A final segmentation is applied after the primitive extraction by the RESC method. From RESC method, the parameters for a given model to the data are obtained. Therefore, the segmentation process sets a threshold based on the fitting process and extracts the largest continuous region where the residual of each pixel is within the threshold limit. A boundary list method is used to perform the segmentation efficiently. It can tolerate one point outliers. Some small regions may occur at the boundary or inside some region. An erosion algorithm is used to eliminate these small regions.

In Chapter 7, various experiments of synthetic and real data in two and three dimensions are presented. Three different methods, RESC method, least median squares method (LMS) and least squares (LS) method, are tested and compared. Among these methods, the RESC method has the best performance in the case of outliers. The breakdown point can be as high as 94% in noise free situations. On average, RESC has much higher breakdown point than LMS method. In the real range image experiments, we tested several range images from National Research Council of Canada (NRC) and from the Pattern Recognition and Image Processing Lab of Michigan State University in public domain. The results are very good.

In Chapter 8, we conclude the dissertation and propose directions for future research.

Bibliography section lists various published articles in journals, conference proceedings, Ph.D. thesis, technical reports, collected books and books on computer vision. Only cited publications in the text are listed. The bibliography list is sorted alphabetically by the first author's last name.

Appendices A, B, D and E contain details of some mathematical derivations. Appendix C shows the derivations of the invariants and pose of quadratic surfaces. Originally, we proposed the idea for object recognition but found the invariants are very sensitive to noise. Therefore, it cannot be used for object recognition. However, we can use it to determine if a surface is a planar or a quadric as explained in Chapter 4. More details can be obtained from our previous publications [85, 82]

Chapter 2

Survey of Range Image Segmentation

In this chapter, we survey various methods for range image segmentation and fitting. Section 2.1 introduces basic concepts and two categories of segmentation. Category one, in section 2.2, contains methods using the properties of surfaces, including the methods based on edges, or regions, or the hybrid of the two. In methods based on edges, we survey the method by Fan, Medioni and Nevatia [21] and the method by Roth and Levine [66, 68]. In region method, we survey region growing method [7, 5, 51, 50], region merging method [23, 70, 78] and clustering method [10, 27, 70, 48]. In hybrid method, we survey the methods of [81, 47]. Another category, in section 2.3, includes methods based on primitive extraction. These methods include: random sample consensus (RANSAC) [24, 8], least median squares (LMS) [69, 66, 68, 55], random Hough transformation (RHT) [80], median of the intercepts (MI) [49], genetic algorithm (GA) [11, 31, 67] and also the residual consensus (RESC) [84, 83, 87] proposed in this dissertation.

2.1 Introduction

A *range image* is a set of three-dimensional coordinates of discrete points on the visible surface of an object. Extracting useful information from these dense points is a crucial task for computer vision systems. Since individual points in a range image provide little information, pixels which share certain intrinsic properties with their neighbors are explored and extracted for higher level processing, e.g. for object recognition. *Segmentation* is the process of finding pixels with similar properties. Formally, let I denote range image, R_i denote the ith region such that pixels in the region have similar properties, the segmentation of I is a partition:

$$\begin{aligned} \cup_{i=1}^{M} R_i &= I \\ R_i \cap R_j &= \phi \quad \text{for } i \neq j. \end{aligned}$$

Each region is normally processed further by classification and surface fitting. The robust and accurate surface fitting is essential to get really useful information from such a partition. Fitting surfaces to pixels in a region is straightforward if a correct and accurate partition is obtained, but how to determine such a partition? On the contrary, if the surface parameters are determined it is not difficult to get an image segmentation, but how to obtain surface parameters? This is often referred to as a "chicken-and-egg" problem [7].

Figure 2.1 illustrates range image and segmentation and fitting process. A laser range sensor scan the object and generate a profile. By moving the range sensor along a line, a whole range image can be generated as show in Figure 2.2.

The methods of segmentation and fitting can be roughly divided into two categories.

1. methods using properties of the surface (segmentation then fitting)
2. methods based on extraction of primitives (fitting then segmentation)

Figure 2.1: Range image segmentation and fitting

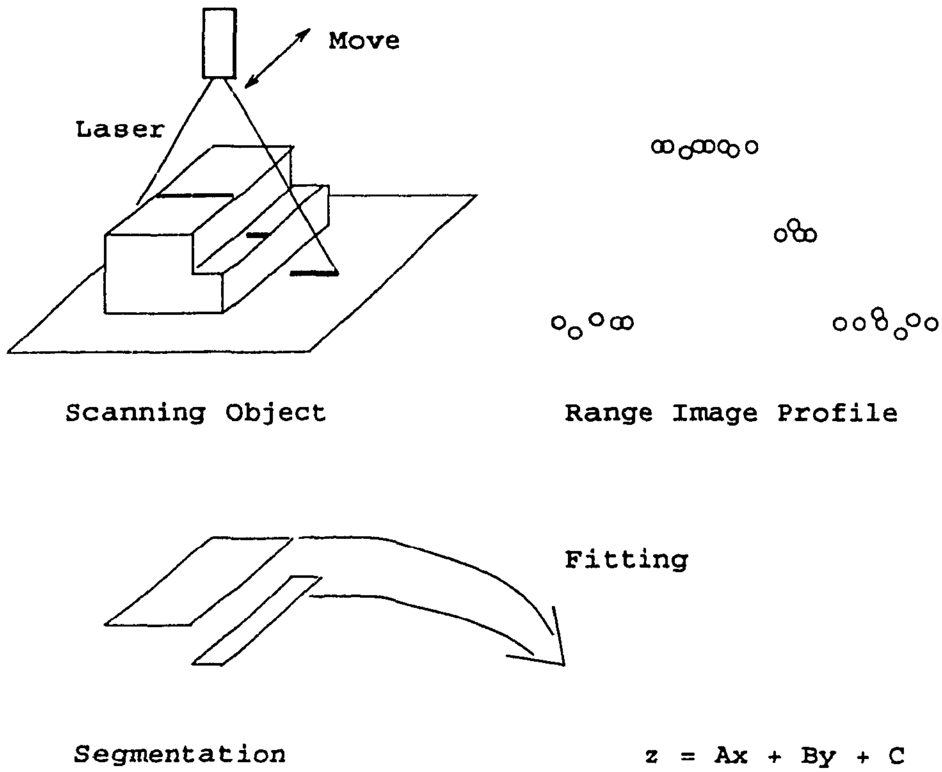
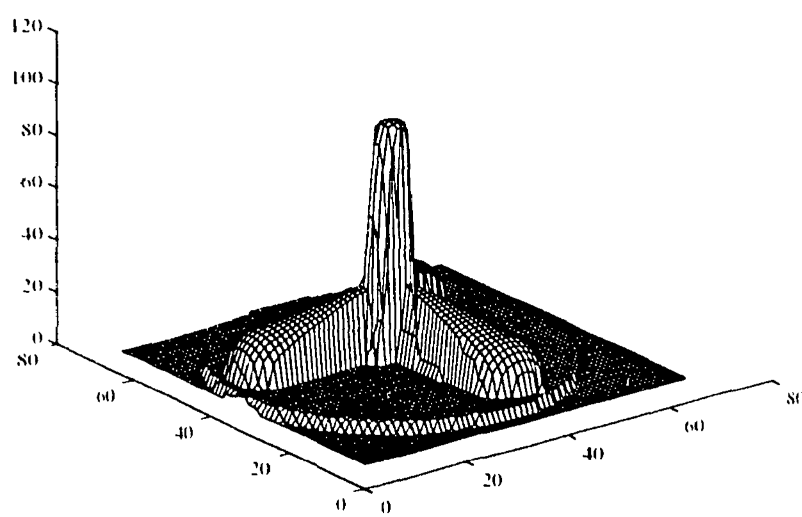


Figure 2.2: A real range image (grip)



The most often used category is the first one. Techniques of the first category explore the properties of each pixel and its neighbors. Normally they perform segmentation by extracting curvatures and then finding edges [21, 66] or regions [7, 5, 51, 50, 23, 78]. Segmentation may also be performed using a clustering method [10, 27, 70, 48], or a hybrid of curvature and clustering [81, 17]. Many papers demonstrate very good segmentation and reconstruction of 3D objects. Each segment should be a *homogeneous* surface region. Most papers consider a homogeneous patch to be a region having the same surface by surface curvature signs, or belonging to the same sign of the surface curvature, or belonging to the same order of surfaces, or a region without discontinuity. The more *meaningful* the segment, the better it is for higher level processing. In [85, 82], we see that object recognition can be performed with high efficiency if each segment is a *quadratic* surface patch. It is difficult to use the first category to extract specific primitives. Therefore, if we want to extract specific primitives, such as quadratic surface patch, from range image, this category is obviously not a good one since the segment is not based on specific primitives. Although the splitting-and-merging is not exactly in the category of segmentation then fitting, it is essentially a region merging method [23, 70, 78]. The whole image is split into the smallest pieces. Fitting and segmentation is performed during the region growing and merging process until the gross error exceeds a threshold. Splitting and merging is not widely used for range image segmentation probably because of the extensive computation for the repeated fitting test.

Methods in the second category extract required primitives directly from the unprocessed range image. The Hough transform (HT) [13, 18] is widely used for extraction of primitives and motion determination [1, 19, 15]. The HT requires a very large space to store parameter voting in order to find primitives according to the maximum vote. To avoid the space requirement, Xu, Oja and Kultanen [80] propose a new curve detection method, the randomized Hough transform (RHT). Liang [51] proposes a curve fitting Hough transform (CFHT). But all these methods need to discretize either the input data or the parameter space. They have problem with finely discretized z values of range image and with the nine parameters required to describe quadratic surfaces.

Recently, *robust estimation* techniques have gained importance in computer vision applications [56]. Robust estimation means that surface fitting is not influenced by *outliers (gross errors) in the processing region*. Fischler and Bolles [21] propose a random sample consensus (RANSAC) paradigm for model fitting to images. The RANSAC method depends exclusively on a predefined threshold. The results are sensitive to this threshold and therefore some knowledge of the scene must be obtained in advance. Besides, in a given scene, different segments may have different standard deviations and require different thresholds. RANSAC cannot handle this problem. Rousseeuw and Leroy [69] propose the least-median-squares method (LMS) which can tolerate 50% outliers. Roth and Levine [66] use LMS for surface fitting. The application of LMS to noisy piecewise constant data with a large proportion of outliers can fail [58]. In [66], segmentation is based on jump edge and roof edge extractions. This method cannot handle smoothly connected segments because it only detects jump and roof edges. It is easier to select a threshold for jump and roof edges using an iterative robust fitting method. Kamgar-Parsi and Netanyahu [49] fit a straight line to a noisy image using a median of the intercepts (MI) method. All these robust estimation methods provide a way to extract primitives from raw data directly, but most of these papers only attempt to extract primitives in 2D images and do not demonstrate complete 3D range image segmentations. Yu, Bui and Krzyżak [84, 83, 86] improve LMS method and demonstrate a complete segmentation of range images.

Segmentation and fitting are difficult tasks. A very good review of this area before 1986 can be found in [6, 11]. The following is a brief review of range image segmentation in recent years.

2.2 Methods Using Properties of Surfaces

Methods belonging to this category use some kind of *properties* of each pixel and its surrounding neighbors to segment the whole image into non overlapping regions in which the pixels have the same properties. The criteria for segmentation can be roughly divided into three classes: (1) extract *edges* among regions by exploring the *discontinuities* [21, 66, 17, 10], (2) *classify* and *grow*, or merge each region until the whole image is segmented [5, 7, 51, 50, 10, 27, 18, 23, 78], (3) a hybrid of the two [81, 17].

2.2.1 Segmentation Based on Edges

Edge is the *discontinuities* between two surface patches and obviously is a good criterion for segmentation. *Jump edges* are the discontinuities of depth values and can be easily detected. *Crease edges* are the discontinuities of surface normals and the detection depends on a correct selection of thresholds. For smooth connected surface patches, there are no obvions edges between them. It seems impossible for this class of method to segment such smoothly connected regions.

Fan, Medioni and Nevatia [21] segment range image by *edges*. First, distinguished points comprising the edges of segmented surface patches are extracted using the *zero-crossings* and *extrema* of curvature along a given direction. Two different methods are used: if the sensor provides relatively noise free range images, the principal curvatures are computed at only one resolution, otherwise, a *multi scale* approach is used and curvature is computed in four directions 45° apart to facilitate interscale tracking. These points are then grouped into curves of the following type

- type 1: isolated positive extremum (+).
- type 2: isolated negative extremum (−).

- type 3: associated positive extremum and zero-crossing (+0).
- type 4: associated negative extremum and zero-crossing (−0).
- type 5: associated positive extremum and zero-crossing and negative extremum (+0−).

These curves are classified into different classes corresponding to significant physical properties such as jump boundaries (+0−), folds (+0, −0, +, or −), and ridge lines (or smooth extrema). Then jump boundaries and folds are used to segment the surfaces into surface patches. This 1D *derivative approach* can provide good localization, but is more sensitive to noise than the 2D window derivative [5]. The method explores intensively the discontinuities between surface patches, but does not consider the homogeneous property of the surface patches. Hence it is very difficult to segment smooth connected regions.

Once the closed boundaries have been obtained, surface patches are approximated by a *bivariate polynomial* (biquadratic):

$$g(x,y) = a_{00} + a_{10}x + a_{01}y + a_{20}x^2 + a_{11}xy + a_{02}y^2$$

The coefficients are obtained by minimizing the least-squares error between observed and interpolated data.

Roth and Levine [66, 68] propose a method for segmentation based on *robust* surface fitting with the *least median squares* [69] technique (LMS, reviewed in the later section). Jump edges and roof edges are extracted to form initial segmentation of images. Connected regions are fitted with LMS method. The fitting is successful if the LMS error is less than or equal to a noise *threshold* F. If this is true the inlier pixels, belonging to the geometric primitive, are assigned to the primitive and removed from further consideration. If the fitting is not successful, then *another set* of jump and roof edge *thresholds* is then chosen and the process is repeated. Finally, if for this particular geometric primitive (say a plane) there is no success at any set of edge thresholds, then the *next* more complex geometric primitive (a quadric in this case) is considered in the same fashion. The entire process is iterated until no more points are left in the image. This method cannot handle smoothly connected segments because only thresholds are used to detect jump and roof edges.

2.2.2 Segmentation Based on Region

Instead of looking for discontinuities among surface patches, the methods in this category explore the properties of each region and classify them. These methods can be divided mainly into three groups, region growing [7, 5, 51, 50], region merging [23, 78] and clustering [10, 27, 70, 18]. Region growing methods and clustering methods try to find a seed of a region by curvature types or clustering, and then grow the region from the seed. Region merging method is sometimes called splitting-and-merging. It is also based on region properties, e.g. Table 2.1. Different from region growing method, the segmentation is performed during the stage of *region merging* process. Edges are then found at the places where two regions meet. It is possible to segment smooth connected regions by these methods.

Region Growing Method

Besl and Jain [7] and Besl [5] published papers and a book on segmentation method. Their publication may be the most intensively study in the area. They classified range images into eight different types based on the signs of *mean and Gaussian curvature*. The initial segmentation is refined by an iterative region growing method based on the variable-order surface fitting.

Mean (H) curvature (average of the maximum and minimum curvature of a point) and Gaussian (K) (product of the maximum and minimum curvature of a point) curvature images are computed on a 7×7 window with equally weighted

	$K > 0$	$K = 0$	$K < 0$
$H < 0$	Peak $T = 1$	Ridge $T = 2$	Saddle Ridge $T = 3$
$H = 0$	(none) $T = 4$	Flat $T = 5$	Minimal Surface $T = 6$
$H > 0$	Pit $T = 7$	Valley $T = 8$	Saddle Valley $T = 9$

H = mean curvature, K = Gaussian curvature and T = surface type label.

Table 2.1: Surface type labels from surface curvature sign

least squares derivative estimation window operators. The surface curvature sign images are then used to determine the surface type image (Table 2.1) and form an initial segmentation based on these surface types. Region seeds are obtained from the initial surface type segmentation by a 3×3 erosion operation (i.e., zero out pixels that have zero valued neighbors and leave other pixels alone). The erosion is repeated inside the initial segment until the remaining number of pixels is less than a threshold.

Iterative variable order surface fitting is then performed starting from the seed regions. The set of approximating functions F ($|F| = 4$) is written in the form of a single equation:

$$f(m, a; x, y) = \sum_{i+j \leq m} a_{ij} x^i y^j.$$

When $m = 4$, the equation is:

$$f(m, a; x, y) = a_{00} + a_{10}x + a_{01}y + a_{11}xy + a_{20}x^2 + a_{02}y^2 + a_{21}x^2y + a_{12}xy^2 + a_{30}x^3 + a_{03}y^3 + a_{31}x^3y + a_{22}x^2y^2 + a_{13}xy^3 + a_{40}x^4 + a_{04}y^4.$$

After a surface of order m^k is fitted to the region R^k in the kth iteration, the surface description is used to grow the region into a larger region where all pixels in the larger region are connected to the original region and compatible with the approximating surface function for the original region. If the magnitude of the residual of a pixel is less than the allowed tolerance value ($w_0\epsilon^k$, where $\epsilon^k = \|$ residual $\|_{R^k}$ and $w_0 = 2.8$), then the pixel is added to the region R^k. Compatibility is checked again for each pixel in the region. If the difference between the normal determined by the given data and the normal determined by the approximating surface is less than a threshold ($\theta_t = 12 + 16\sigma_{img}$ degrees), the pixel is compatible. The largest connected region R^k that overlaps the seed region is then extracted to create the next region R^{k+1}. The process is repeated until $|R^k| = |R^j|$ for any $j < k$ or ϵ^k [illegible] ϵ_t, and m^k [illegible] $|I|$.

Kasvand [51, 50] proposes a method to segment and classify surface patches by their curvatures. The basic parameters for a surface element are its position (x, y, z), its unit surface normal vector $\mathbf{n}(x, y, z)$, and the maximum surface curvature $k1(x, y, z)$ and the minimum surface curvature $k2(x, y, z)$. A surface element has eight degrees of freedom, three for the position, three for orientation in space and two from the $k1$ and $k2$ values. The orthogonal directions of $k1$ and $k2$ as well as their values span to a meaningful $k1$-$k2$ space. From the $k1$-$k2$ space the surface element can be classified as: *flat* if $k1 = k2 = 0$, *cylindrical* and *conical* surface if [illegible] zero, *sphere* if $k1 = k2 \neq 0$, etc. Also, a *natural* edge detector is obtained *automatically* from the $k1$-$k2$ space since at any edge the theoretical $k1$ value is infinite.

The image can be segmented into regions which have become *automatically recognized* according to the type of the surface in the region. The segmentation processing sequence is as follows:

1. Construct two dimensional histogram $H(k1, k2)$. Regularize $H(k1, k2)$, carry out dynamic thresholding and use constraint for cylindrical and extreme regions. Give *identity* to the *isolated* region by region labeling and *spread* the region labels out, since dynamic thresholding only [illegible] the peaks and ridges in $H(k1, k2)$.

2. Extract the planar or flat areas in the vicinity of $k1 = k2 = 0$ in $k1$-$k2$ space.

3. Extract the positive and negative cylindrical and conical regions where the pixels fall onto the $k1$ axis.

4. Extract the extreme curvature edges.

The boundaries between differently labeled regions are created by zeroing masks which has different labels within its 3 by 3 pixel neighborhood. This creates *cracks* between the labeled regions, convert it to a binary, la[illegible]l the connected components (facets), and approximate the facets with suitable analy[illegible]ic functions (planar or bi-quadratic). Compare the analytical value with the actual values to eliminate *wrong* pixels. Extend the coverage towards the neighboring regions and create a *contest* between the competing facets. The process is controlled by updating the labels and iterated until the process *stabilizes*.

Region Merging Method

Reg[illegible] merging method is different from the region growing method. Region growing method is to segment the range image into regions based on curvatures and growing each region from the seed in the initial segmentation. The region merging method is first to segment the image into many regions (over segmented) without exploring properties in each region and then to merge these regions according to some criterion. Region merging method is also called splitting-and-merging method.

The splitting and merging paradigm for curves was first introduced by Pavlidis and Horowitz [60, 61]. The idea is simple. First split the arc into segments for which the error in each segment is sufficiently small. Then try to merge successive segments, providing any resulting merged segment has sufficiently small error. Then try to adjust the breakpoints to obtain a better segmentation. Do this repeatedly until all three steps produce no further change.

Faugeras and Hebert [23] use a splitting-and merging strategy for the segmentation and fitting of range images. The image is first split into small triangles and then the region is grown during the fitting process. They claim that the best solution is to use as global a strategy as possible which means that evolution of the segmentation is determined by the quality of the overall description of the surface. The global control prevents the segmentation from being perturbed by local noisy measurements. Their global control has two consequences:

1. In each iteration, the regions R_i and R_j which produce the minimum error $E(R_i \cup R_j)$ among the whole set of pairs are merged.

2. The program stops when the global error $\sum_{i=1}^{N} E(R_i)$ is greater than E_{max}.

Surfaces consist of planes and quadrics. The error E is defined as the distance between the points of the region and the best fitting plane in the least squares sense. For quadrics, the error measure is not the distance from a point to a quadric surface but:

$$E = \mathrm{Min} \sum_{i=1}^{N} (\mathbf{x}_i^t \mathbf{A} \mathbf{x}_i + \mathbf{x}_i \cdot \mathbf{v} + d)^2$$

where

$$\mathbf{A} = \begin{bmatrix} a_1 & a_4/\sqrt{2} & a_5/\sqrt{2} \\ a_4/\sqrt{2} & a_2 & a_6/\sqrt{2} \\ a_5/\sqrt{2} & a_6/\sqrt{2} & a_3 \end{bmatrix}$$

$$\mathbf{v} = (a_7 \ a_8 \ a_9)^t$$

$$d = a_{10}$$

The constraint $\mathrm{Tr}(\mathbf{A}\mathbf{A}^t) = \sum_{i=1}^{6} a_i^2 = 1$ for function E is used to avoid the trivial solution [0, ..., 0] and the constraint is invariant to translation because $\mathbf{A}$ is invariant and to rotation, because the trace operator is also invariant. The minimum of E is found by using the Lagrange multipliers method. The switch between plane and quadrics is done automatically by simply comparing the respective error. The quadratic surface fitting method used here does not minimize geometric distance between fitted surface and actual data. This kind of fitting is very sensitive to noise [85]. Therefore, fitting and segmentation cannot be accurate.

Sabata, Arman and Aggarwal [70] (reviewed in section 2.2.2) use similar strategy to merge oversegmented image. Bivariate polynomials of up to *fifth* degree are used to *represent surfaces*. Two adjacent surface patches are merged if parameters of one of the patches, when used to extrapolate over the neighboring patch, result only in a small error.

Taubin [78] addresses the problem of parametric representation and estimation of complex planar curves in 2-D, surfaces in 3-D and nonplanar space curves in 3-D, and proposes a segmentation method using the methods mentioned above. The representation of curves and surfaces is in implicit form $Z(f) = \{x : f(x) = 0\}$, where $f : \Re^n \rightarrow \Re^k$ is a smooth map, a map with continuous first- and second-order derivatives at every point, and

$$f_1(x) = 0, \quad \ldots, \quad f_k(x) = 0.$$

$Z(f)$ is a planar curve if $n = 2$ and $k = 1$, it is a surface if $n = 3$ and $k = 1$, and it is a space curve if $n = 3$ and $k = 2$. The approximate distance from x to $Z(f)$ is:

$$\sqrt{f(x)(Df(x)Df(x)^t)^{-1}f(x)}$$

where $Df(x)$ is Jacobian of $f(x)$. In the case of planar curves and surfaces, $k = 1$ and the approximate distance from x to $Z(f)$ is $\sqrt{f(x)^2/||\nabla f(x)||^2}$, which has been widely used for curve fitting. The minimization of the approximate mean square distance is known as the nonlinear least squares problem, and can be solved using iterative methods, such as the well-known Levenberg-Marquardt algorithm. A good initial estimation is necessary for the Levenberg-Marquardt algorithm. In the linear case and the cases of circles, spheres and cylinders, the minimization problem reduces to the generalized eigenvector fit, which minimizes the sum of squares of the values of the functions that define the curves of surfaces under a quadratic constraint function of the data. The generalized eigenvector fit is independent of the choice of coordinate system, which is a very desirable property for object recognition, position estimation, and the stereo matching problem. A reweight procedure is introduced to improve the solution produced by the generalized eigenvector fit at a lower cost than the general iterative minimization techniques. Finally, the result of the reweight procedure is fed into the Levenberg-Marquardt algorithm to minimize the approximate mean square distance.

The segmentation algorithm is partially based on Besl and Jain's variable order surface fitting algorithm [7, 5] and Silverman and Cooper's surface estimation clustering algorithm [71], and related to Chen's planar curve reconstruction algorithm [10]. The square of the noise variance at one data point $\sigma(x)$ is estimated by fitting a straight line or plane to the data in a small neighborhood of the point, which is a circle or ball of radius equal to a few pixels, using the eigenvector fit method and then computing the approximate mean square distance to the fitted line or plane. The square of the noise variance at every data point is estimated and a histogram of them is built. The data points with square noise variance in the top 10% of the histogram are marked as outliers. The goodness of fit test is a two step test: (1) approximate mean square distance $\Delta_{\mathcal{D}}^2(\alpha)$ test (related to χ^2 statistic)

$$c_1\hat{\sigma}_{\mathcal{S}}^2 < \Delta_{\mathcal{D}}^2(\alpha) < c_2\sigma_{\mathcal{S}}^2$$

where $0 < c_1 < 1 < c_2$ and

$$\sigma_{\mathcal{S}}^2 = \frac{1}{q}\sum_{i=1}^{q}\sigma^2(p_i)$$

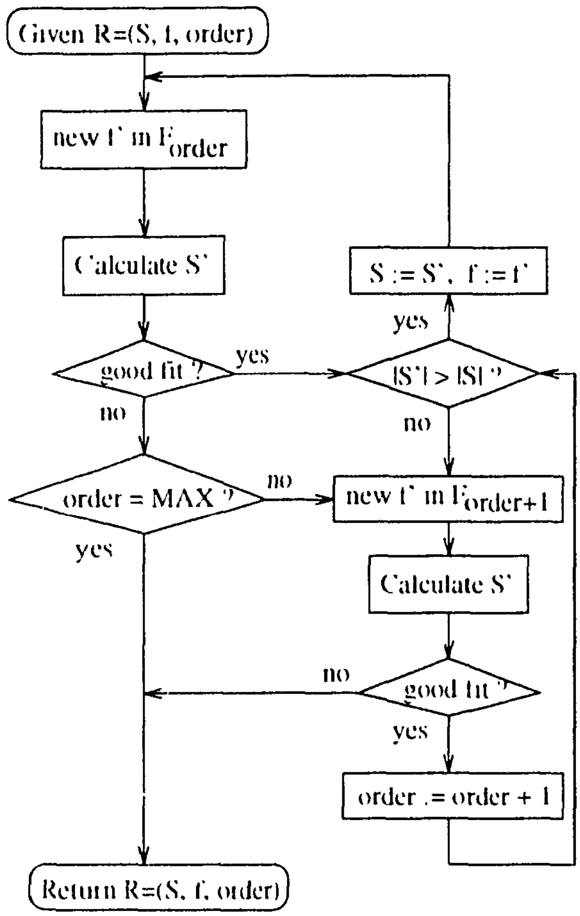
is the mean noise variance estimate on the set $\mathcal{S}$, (2) the second test

$$\delta_{\mathcal{S}}^2(\alpha) < c_3\Delta_{\mathcal{D}}^2(\alpha)$$

where $c_3 > 1$ is another test constant.

The variable order region growing algorithm can be described as follows. An increasing sequence $\mathcal{F}_1 \subseteq \cdots \subseteq \mathcal{F}_{order_{max}}$ of families of functions is given. A *region* is a data structure $R = (\mathcal{S}, f, order)$, where $\mathcal{S}$ is a connected subset of data points and f is an element of $\mathcal{F}_{order}$ that approximates every point of $\mathcal{S}$ well. The region

Figure 2.3: The flowchart of Taubin's algorithm



growing starts by finding a *seed region* $\mathcal{R} = (\mathcal{S}, f, 1)$, where f is an element of $\mathcal{F}_1$, whose set of zeros $Z(f)$ approximates every point of $\mathcal{S}$ well. In this case $\mathcal{F}_1$ is the family of first degree polynomials, and a seed region is the subset of data points in the neighborhood of a point not marked as an outlier, which was used to estimate the noise variance at the point, together with the fitted straight line or plane.

Then, given a current region $\mathcal{R} = (\mathcal{S}, f, order)$, the following loop is repeated until no further growth in $\mathcal{S}$ is observed. The maximal connected region $\mathcal{S}'$ of points well approximated by f and intersecting the initial seed set is computed. If $\mathcal{S}'$ does not have more points than $\mathcal{S}$, neither $\mathcal{S}$ nor f is changed, and the loop is exited.

Otherwise, a new member f' of $\mathcal{F}_{order}$ is fitted to $\mathcal{S}'$, and if it satisfies the goodness of fit test, the region $\mathcal{R}$ is replaced by $\mathcal{R} = (\mathcal{S}', f', order)$ and the loop repeated. If f' does not satisfy the test, the loop is exited. When the loop is exited, if $order$ is equal to the maximum order $order_{MAX}$, the region growing is finished by returning the current region $\mathcal{R} = (\mathcal{S}, f, order)$. Otherwise, a member f' of $\mathcal{F}_{order+1}$ is fitted to $\mathcal{S}$, and if it satisfies the goodness of fit test, f is replaced by f', $order$ is replaced by $order + 1$, that is, $\mathcal{R}$ is replaced by $\mathcal{R} = (\mathcal{S}, f', order + 1)$, and the loop is traversed once more. If f' does not satisfy the test, the region growing is finished by returning the current region $\mathcal{R} = (\mathcal{S}, f, order)$. The flowchart of this algorithm is given in Figure 2.3.

Taubin's fitting algorithm of minimization of the approximate distances is an improvement of the method used by Faugeras and Hebert [23]. The solution involves nonlinear iterative optimization method. The computation is extensive for such iterative algorithm. Therefore, in most cases, a simplified fitting is used in the segmentation algorithm.

Clustering Method

Hoffman and Jain [10] and Flynn [27] propose a segmentation method based on clustering technique. A pattern is defined as $\mathbf{c_i} = (x_i, y_i, z_i, n_{x_i}, n_{y_i}, n_{z_i})$ where x_i, y_i, z_i are coordinates of point i and $n_{x_i}, n_{y_i}, n_{z_i}$ are the unit normal vector of point i. The surface normal of each point is extracted from the parameters of a fitted plane to the image data in a neighborhood of the considered pixels. A cluster center is the centroid of the patterns assigned to that cluster. The labeling obtained from a k cluster solution is denoted $\{l_1, \ldots, l_n\}$ and the k cluster centers are $\mathbf{m}_1, \ldots, \mathbf{m}_k$ with $\mathbf{m}_k = (m_{k1}, \ldots, m_{k6})$. The squared error is given by,

$$E^2 = \sum_{i=1}^{n} \sum_{j=1}^{6} (c_{ij} - m_{l_i j})^2$$

where l_i is the class label obtained for the ith input pattern $\mathbf{c}_i$. The first clustering places all patterns in one cluster. k_{max} is an *a priori* upper bound on the number of clusters. The initial segmentation based on CLUSTER program is as follows:

- A clustering with $j+1$ clusters is obtained from a clustering with j clusters by choosing the pattern farthest from the current clustering as the new cluster center.

- If any two clusters in that solution can be merged to produce a $(k_{max}-1)$-cluster solution with lower squared error than the previous $(k_{max}-1)$-cluster solution, the resultant clustering replaces the previous $(k_{max}-1)$-cluster solution. Then the $(k_{max}-1)$-cluster clustering is examined in the same way to produce a (possibly new) $(k_{max}-2)$-cluster clustering, and so forth.

Steps 1 and 2 are repeated sequentially until none of the k_{max} clusterings change during a pass. $k_{max} = 16$ [27] and $k_{max} = 20$ [10] in the program.

Hoffman and Jain [10] use different merit function. The average within-cluster interpoint distance of cluster i defined as:

$$CLAVGD(i) = \frac{1}{|G_i|}\sum_{x\in G_i} d(x, c(i))$$

where d indicates Euclidean distance, $c(i)$ is the center of cluster i, and G_i is the set of points belonging to cluster i. A statistics $M(i)$ is defined to reflect the isolation and compactness of cluster i:

$$M(i) = \frac{\min_{j, j\neq i} d^2(c(i), c(j))}{CLAVGD(i)}.$$

An overall merit function M_{ave} is defined as a weighted average of $M(i)$'s, where the weights are the numbers of pixels in clusters. Those clusters with larger values of M_{ave} are preferred over those with smaller values.

The initial segmentation is often contaminated by the following undesirable artifacts:

- Non-connected patches with the same label.

- Patches with an extremely small number of pixels relative to the other image segments.

- Differently-labeled patches belonging to the same object surface.

The refinements of the initial segmentation are performed in three steps to remedy the above problems. A recursive connected component labeling algorithm is used to make the label unique and preserved 8-connected regions with pixel number exceeding a threshold. If the changes of surface orientation are below a threshold value, the patches are merged.

Surface classification is based on fitting errors and curvatures. A plane is fit to the 3D points in the segment, and accepted if the mean squared error of the fit is below an empirically-determined threshold. If the threshold is exceeded, based on curvature analysis, the segment will be fitted to a cylindrical patch or spherical patch. If a squared-error statistic exceeds a threshold, the patch is labeled "unknown" and no more attempt will be performed on that segment. A nonlinear optimization technique is then used to refine the parameters (e.g., radius, orientation) of the resulting surface [26].

Sabata, Arman and Aggarwal [70] propose a segmentation method based on clustering using *pyramidal* data structures. The initial clustering is performed using four properties calculated by the preprocessing stage for each point in the range image. The four properties are the surface normal vector and its three projection onto the xy-plane, the yz-plane and the xz-plane. In pyramidal clustering, the pixels with similar properties are clustered into groups in a hierarchical manner. The pyramidal algorithms are divided into three stages. The first process is initialization where the nodes of the pyramidal data structure are initialized. The base level is initialized by assigning the pixel values of the image to the corresponding nodes. The other levels of the h level pyramid are initialized by taking the average of a area in level $l-1$ to generate a node in level l. Each node is the clustering of the lower level nodes. The second stage is the *node linking*, each node *chooses* its best father based on a closeness measurement. The last step is the tree generation, using the results of the linking, and it assigns a region label to each node. Starting from level $H \leq h-1$, a distinct label is assigned to all nodes and their children with distinct property values. An over-segmented performance is assumed and a high level merging is necessary. Bivariate polynomials of up to *fifth* degree are used to represent surfaces. Two adjacent surface patches are merged if parameters of one of the patches, when used to extrapolate over the neighboring patch, result only in a small error.

Johon, Meer and Bataouche [18] propose a robust clustering technique based on the robust minimum volume ellipsoid estimator (MVE) of Rousseeuw and Leroy [69]. The algorithm has an iterative nature. Let $\mathbf{X}$ be a set of n distinct data points (feature vectors) in a p dimensional feature space:

$$\mathbf{X} = \{\mathbf{x}_i, \quad i = 1, \ldots, n\} \qquad \mathbf{x}_i = (x_1^i \ldots x_p^i)^t.$$

Every point has associated with it a scalar positive weight q_i. Denote by $\mathbf{X}_l$ the set of data points contained in the feature space at the l-th iteration. From this set, the best cluster (to be characterized below) is delineated and removed yielding the new set $\mathbf{X}_{l+1}$. The process stops whenever the number of remaining points becomes less than the assumed minimum cluster size or the number of detected clusters exceeds an upper bound. To extract a cluster, the space $\mathbf{X}_l$ is analyzed at different "resolutions" characterized by a step size h, $h \leq 0.5$. For a given value of h, they seek the minimum volume ellipsoid containing fraction h of the mass of $\mathbf{X}_l$, $Q_l = \sum_{\mathbf{X}_l} q_i$. A random sampling method is used. The feature spaces contain around 100 data points. After the first point is chosen at random, the remaining p points are chosen from a box centered around it. The number of samples per iteration is 25. A cluster is delineated based on this ellipsoid and its shape is compared with the shape of an ideal cluster generated by a Gaussian density. The cluster yielding the smallest significance level over all h values is the best cluster in $\mathbf{X}_l$. They call the algorithm the generalized minimum volume ellipsoid (GMVE) clustering since it employs several values of h, whereas the original MVE estimator has $h = 0.5$ and all $q_i = 1$ (least median squares method).

A p-tuple of facet parameters is mapped into the feature space for clustering processing. The parameters are estimated at locations situated l pixels apart along either coordinate axes. The input range image is tessellated with $(2l+1) \times (2l+1)$ windows with almost 50% overlap. The facet parameters β_k are obtained by a robust *M-estimators* [11] as iteratively reweighted least squares with the definition of the weights depending on $\rho(r_{u,v}) = r_{u,v}^2$. The weight function (also known as Tukey's biweight) is:

$$w_{u,v} = \begin{cases} [1 - (\frac{r_{u,v}}{c\hat{\sigma}_{u,v}})^2]^2 & |r_{u,v}| \le c\hat{\sigma}_{u,v} \\ 0 & \text{otherwise} \end{cases}$$

where $\hat{\sigma}_{u,v}$ is the locally estimated standard deviation of the fit and c is a tuning constant taken equal to 4.685 to assure superior performance for the Gaussian noise [12]. At the end of the clustering algorithm, not all the feature points were allocated to clusters, and the number of unlabeled feature points can exceed the minimum accepted cluster size. In post process of the segmentation first "seed regions" are delineated containing the pixels that can unequivocally be mapped, that is the absolute valued residual must be less than 2.5 times the global standard deviation estimate of the noise. The remaining pixels are incorporated through region growth. At every expansion step, a one-pixel-wide ring along the perimeter of each region is examined. If the difference between the estimated fit and the pixel value is less than 2.5σ the pixel is incorporated into the expanding region. The expansion process stops at the collision of two regions or when no more pixels can be conquered. In the experiments, the window size is 7×7, $l = 3$ for planar facet model and 15×15, $l =$ [illegible] for biquadratic facet model. When the homogeneous regions have small size the cluster detection becomes unreliable [18].

2.2.3 Hybrid of edge-based and region-based method

Yokoya *and* Levine [81] propose a hybrid approach to the image segmentation problem. The range image of 3-D objects is divided into *surface primitives* which are homogeneous in their intrinsic differential geometric properties and do not contain discontinuities in either depth or surface orientation. The method is based on the computation of partial derivatives which are obtained by a selective local biquadratic surface fit. Then by computing the Gaussian and mean curvatures, an initial region-based segmentation is obtained in the form of a curvature sign map. Two additional *initial edge based segmentations* are also computed from the partial derivatives and depth values: *jump* and *roof edge* maps. The three image maps are then combined to produce the final segmentation.

Jain and Nadabar [17] propose a hybrid segmentation method which combines the initial region-based segmentation of Hoffman and Jain [10] and Markov Random Field (MRF) model based *boundary detection method*. The *jump and edge* likelihoods at each edge site are computed using special local operators. These likelihoods are then combined in a Bayesian framework with a MRF prior distribution on the edge labels to derive the posterior distribution of labels. An approximation to the maximum a posteriori estimate is used to obtain the edge labeling. The edge detection method, like all other edge based segmentation methods, does not always result in closed boundaries. To overcome this problem, they use the region based segmentation algorithm to obtain an initial oversegmented solution. The boundary segments in the oversegmented solution are validated using evidence from the edges found in the MRF based edge detection algorithm.

2.3 Methods Based on Extraction of Primitives

Methods in this category depend on *robust estimation* method. A method is *robust* if primitive parameter estimation can toleration *outliers*. Outliers are normally defined as a very small region inside a surface patch where *residuals* are much higher than the normal noise level of the patch. Here we extend the concept of outliers. Outliers are the pixels which have residuals much larger than that of most pixels in the current region. Outliers may not be just a small region, they may be another surface patch and in some circumstances the number of outliers exceeds the number of pixels in one surface patch of the current region. Therefore, fitting raw range images is the processes to find the largest homogeneous region which is the expected primitive and to consider all others as outliers. A *breakdown point* is the percentage of outliers that can be tolerated before breakdown occurs. The traditional least squares algorithm has a breakdown point of 0% because one outlier may cause the result failure.

2.3.1 Random Sample Consensus (RANSAC)

Fischler and Bolles [21, 8] propose a random sample consensus (RANSAC) paradigm for model fitting to images. RANSAC is the first method to use a *random sampling* approach for surface fitting. The most important advantage is that the method is not sensitive to *outliers* (or gross errors). Least squares approach cannot filter out outliers. The RANSAC paradigm [21] is as follows:

1. Given a model that requires a minimum of n data points to instantiate its free parameters, and a set of data points P with more than n points. Randomly select n data points from P and instantiate the model. Determine the subset S (consensus set) of points in P that are within some error tolerance of the model.

2. If cardinality of subset S is greater than some threshold T, which is a function of the estimate of the number of gross errors in P, use S to compute (possible using least squares) a new model.

3. If cardinality of subset S is less than T, *randomly* select a new subset S and repeat the above process. If, after some predetermined number of trials, no consensus set with T or more members has been found, either solve the model with the largest consensus set found, or terminate in failure.

To evaluate the quality of the fit, Bolles and Fischler [8] use:

> Error Tolerance-Test: The percentage of residuals that lies within a context dependent tolerance band.
>
> Sign-Test: The ratio of positive to negative residuals.
>
> Run Length Test: The length of the longest sequence of monotonically increasing or decreasing residuals.

The error tolerance test provides the primary basis for accepting or rejecting a model. The sign and run-length tests are perfectly general in that they require no problem dependent information, but are obviously weaker and thus can provide only secondary evaluation criteria. In order to use RANSAC, one has to predefine the error tolerance and threshold T. These are the key parameters to make it work properly because at different noise levels or in different models, the parameters should be different. But it is not easy to select the correct ones.

2.3.2 Least Median Squares (LMS) Method

Rousseeuw and Leroy [69] invent the *least median squares* method (LMS) which can tolerate 50% outliers. LMS algorithm can be used to obtain a *robust* fitting. The algorithm can be described by the case of line fitting to a set of N points. Two points are required to define a line uniquely. The algorithm *randomly* selects K sets of two points. For the line defined by each set of two points, the *residuals* (errors) of all the N points related to this line are computed and squared. Then the *median* of these squares is found (the median is the *middle* element of the sorted squared residuals) and associated with this particular set. The set which has the least median squared (LMS) error is the representative set for the line and the standard deviation of the line fitting to the inlier part can be calculated from the median squared residual. The outliers can be discarded from N points by a *threshold*. Various primitives can be fitted with LMS method, such as planes, quadrics, *etc.* Roth and Levine [66, 68] use LMS to fit primitives to the initial segmentation and then adjust the edge extraction threshold according to the fitting results (reviewed in section 2.2.1). Meer and Mintz [55] also use LMS for *robust estimation* in computer vision. They demonstrate the segmentation of gray level image based on this LMS method. The application of LMS to noisy piecewise constant data with a *large* fraction of outliers can result in *failure* [58]. Also since the sorting of the residual is necessary to find the *mode* of the probability distribution of residuals [55], the complexity of the algorithm is at least $O(n \log_2 n)$ for just the sorting part. In [69], Rousseeuw and Leroy prove a theorem that the 50% breakdown point is the best for a robust estimation method to achieve. But Yu, Bui and Krzyżak [81, 83, 86] introduce RESC method which has a breakdown point more than 50%. In [68], Roth and Levine also demonstrate that the 50% breakdown points can be surpassed with a modified LMS method. Instead of taking the middle of the residuals, they take the lth position of the sorted n residual as the criterion for the random sampling method. The selection of l is more or less arbitrary.

2.3.3 Random Hough Transformation (RHT)

The *Hough transform* [13, 18] is a method for detecting straight lines and curves on gray level images. Given the family of curves being sought, the method produces the set of curves from that family that appears on the image. Stockman and Agrawala

76] were the first to realize that the Hough transform is template matching. Rosenfeld [65] describes an implementation that is almost always more efficient than the *original Hough formulation*. The Hough transform method is extensively surveyed by Illingworth and Kittler [15]. We do not survey it here.

Xu, Oja and Kultanen [80] propose a new curve detection method: *randomized Hough transform* (RHT). For a curve with n parameters, instead of transforming every pixel into a hypersurface of the n-D parameter space as the HT and its variants do, they randomly pick n pixels and solve the required equation to get parameters for the selected primitives. Map the parameters onto one point in the parameter space and increase the counter at that point by one. A primitive is claimed to be found if the counter value of some point in the parameter space exceeds a predefined threshold. The authors claim that the method has the advantages of small storage, high speed, infinite parameter space and arbitrarily high resolution. The examples in the paper are 2D binary images on discrete grid. The values for x and y coordinates are discrete values. Therefore, the resolution of the parameter space is actually constrained by the resolution of the coordinates. It is very difficult to get convergence in accumulators of the parameter space if neither the coordinate values nor the parameter spaces are constrained to discrete levels. Therefore, RHT method has problems with finely discretized z values of range image and with the nine parameters required to describe quadratic surfaces.

2.3.4 Median of the Intercepts (MI)

Kamgar-Parsi and Netanyahu [19] propose *median of the intercepts* (MI) method to fit a straight line to a noisy image. The equation of the line is

$$\frac{x}{a} + \frac{y}{b} = 1.$$

The parameters a and b are the x axis and the y axis intercepts, respectively. From a pair of points a line equation can be solved and the corresponding intercepts can be obtained. For N points, there are $L = N(N-1)/2$ lines altogether, which provide (at most) L pairs of estimates for the intersections. The median estimate of the intercept a (or b) is the median of the entire set $\{a_{ij}\}$ (or $\{b_{ij}\}$), that is

$$a = \text{median}\{a_{ij}\} \qquad b = \text{median}\{b_{ij}\}$$

where $\{a_{ij}\}$ and $\{b_{ij}\}$) are the intercepts of the line passing through the points i and j. Similarly to LMS method, MI can tolerate 50% outliers. Complete combinatorial search makes the method difficult to handle large number of data points.

2.3.5 Genetic Algorithm (GA)

Roth and Levine [67] incorporate GA in the primitive extraction algorithm. Genetic algorithms are a class of global optimization techniques that earn their name from a similarity to certain processes that occur at the interactions of biological genes. Basically, a genetic algorithm selects high strength *parent* models, forming *offsprings* by recombining components from the parent models. The offsprings replace weak models in the system and enter into further competitions. Genetic algorithms have been studied intensively by Holland [41], Goldberg [34] and others.

A concept of *minimal subset* is emphasized in the paper. The minimal subset is the minimal number of points necessary to define different geometric primitives. A minimal subset of the points described by a geometric primitive is often a good representation of the primitive. Instead of using parameter vector **P** for GA operation, the minimal subset **X** with p points is used as gene structure. The GA takes two randomly chosen individuals (parents) and applies a cross-over operation to re-arrange the points of their parents, followed by a mutation operation to take new points from input data, to create two new population members (children). To score each individual in the population, a fixed band method is used. The score is simply the total number of input points contained in the fixed band around each geometric primitive, since the more points belonging to it, the less likely that the alignment of points is random, and the better chance that these points truly belong to the geometric primitive. A steady state GA is used, along with a uniform crossover operation. The initial *population* size is 50, the crossover probability is 0.8 and the mutation probability is 0.05.

2.3.6 Residual Consensus (RESC)

Yu, Bui and Krzyżak [84, 83, 87] propose RESC method. Genetic algorithm [67] is incorporated into the method to accelerate the random sampling speed. RESC can tolerate more than 80% outliers. The residual consensus is measured by a compressed histogram method which has time complexity of $\mathcal{O}(n)$, better than LMS's $\mathcal{O}(n \log n)$, and can handle widely varied noise levels. The RESC method is presented in Chapters 3, 4, 6 and 7 of this dissertation. RESC in genetic algorithm (GA) to accelerate the search speed. Chapter 5 explains how a genetic algorithms works and what is the best GA parameter settings.

2.4 Summary

Segmentation is a fundamental and active research area in computer vision. Various methods and approaches are coming out continuously. It is difficult to say which method is the best since every method has its advantages and disadvantages, applicability, restrictions, *etc.* Because segmentation is normally the first step in computer vision processing, the criterion for segmentation algorithm depends on requirements of the higher level processing. Generally we can say that the better the low level segmentation, the easier the high level processing. If the low level provides accurate and easy to use segmentation and surface parameters, the high level processing is relatively easier and faster. But such low level processing may be difficult and takes a long time. On the other hand, if the low level processing is fast, but provides inaccurate results, the high level processing may be difficult and time consuming. A well developed vision system should consider various factors and perform well on the whole. Therefore, the final criterion for evaluation of the segmentation should be combined with the evaluation of the whole vision system. But whatever the criterion is, faster, more accurate and easy-to-use segmentation methods are always needed. The RESC algorithm is a leg in that direction.

Chapter 3

Estimation Analysis

Estimation analysis is an important statistical tool with applications in most sciences. In section 3.1, we explain the basic definitions and concepts of estimation analysis. In section 3.2, we give the formal definition of outlier and breakdown point which are the important measures for a robust estimation method. In section 3.3, we analyze the problem of second order primitive fitting with least squares method, and compare the different results of surface or curve fitting by the minimization of algebraic distance and geometric distance.

3.1 Estimation Model

The purpose of estimation analysis is to fit equations to observed data sets. The classical linear estimation model is:

$$y = \mathbf{x}\boldsymbol{\theta}, \tag{3.1}$$

where $\mathbf{x}$ is a *model frame*, also called the *explanatory variables* or *carriers* [60]

$$\mathbf{x} = [x_1 \quad \dots \quad x_p]. \tag{3.2}$$

$\boldsymbol{\theta}$ is vector of the estimation coefficients

$$\boldsymbol{\theta} = \begin{bmatrix} \vartheta_1 \\ \vdots \\ \vartheta_p \end{bmatrix}. \tag{3.3}$$

and y is called *response variable*. An instance of $\mathbf{x}$

$$\mathbf{x}_i = [x_{i1} \quad \dots \quad x_{ip}] \tag{3.4}$$

will have a response value of y_i:

$$y_i = \mathbf{x}_i\boldsymbol{\theta} \tag{3.5}$$

A estimation analysis is to find an estimation of $\boldsymbol{\theta}$, i.e., the best fit by some criterion such as the least squares of errors, to n sets of instances:

$$y_i = \mathbf{x}_i\boldsymbol{\theta} + e_i \qquad \text{for} \quad i = 1, \dots, n, \tag{3.6}$$

where $n \geq p$ is the sample size and e_i is the *error term* assumed to be normally distributed with mean zero and unknown standard deviation σ in the classical theory. If $n = p$, Equation (3.6) has a unique solution. If $n > p$, Equation (3.6) is an overdetermined system. Equation (3.1) with estimated $\boldsymbol{\theta}$ can be used to obtain an estimate of y:

$$\hat{y}_i = \mathbf{x}_i\hat{\boldsymbol{\theta}}. \tag{3.7}$$

where $\hat{y}_i$ is called the *predicted* or *estimated* value of y_i. The *residual* r_i of ith instance is defined as the difference between what is actually observed and what is estimated

$$r_i = y_i - \hat{y}_i \tag{3.8}$$

The most popular estimation estimator is the *least squares* (LS) method which minimizes the sum of squared residuals:

Table 3.1. Model format for geometric models

Model type	$\mathbf{x}$	y
line	$1 \quad x_1$	x_2
circle	$x_1^2 \quad x_2^2$	1
conic	$x_1^2 \quad x_2^2 \quad x_1x_2 \quad x_1 \quad x_2$	1
plane	$1 \quad x_1 \quad x_2$	x_3
sphere	$x_1^2 \quad x_2^2 \quad x_3^2 \quad x_1 \quad x_2 \quad x_3$	1
quadratic	$x_1^2 \quad x_2^2 \quad x_3^2 \quad x_1x_2 \quad x_1x_3 \quad x_2x_3 \quad x_1 \quad x_2 \quad x_3$	1

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} r_i^2. \tag{3.9}$$

The linear estimation model does not restrict the estimation equation to a linear one. The fitting model is problem dependent. The model can be lines, circles, conic curves *etc.* in 2D geometric analysis, or, planes, spheres, ellipsoid, quadratic surfaces *etc.* in 3D geometric analysis, or some other model in other cases. The formats for different geometric models are listed in Table 3.1, where x_1, x_2 and x_3 represent x, y and z, respectively, in a real coordinate system. In some of the models, the first element of $\mathbf{x}$ is set to 1 to obtain a constant term in the estimation equation. In general, taking a carrier identical to 1 is a standard trick used to obtain estimation with a constant term. In the second order models, the y term in equation (3.1) is set to 1. This is another standard trick to estimate non-linear equation coefficients by a linear estimation model [37]. If the y term is set to a constant, the residual defined in equation (3.8) in this case has a different geometric meaning. We will explain it in the section 3.3. In the algorithm described in chapter 4, we will use another definition of residual which has a geometric meaning. In this dissertation, we mainly deal with the estimation of two dimensional or three dimensional geometric primitive models.

3.2 Outlier and Breakdown Point

An outlier is the point which is far away from most other points. The *breakdown point* is the percentage of outliers that may force the estimation arbitrarily out of meaningful range. These concepts were first introduced by Hodges [16] in 1967. Later, Donoho and Huber [17] introduced a finite sample version of the breakdown point definition. Here, we adopt[1] their definitions of an outlier and a breakdown point. The formal definition is expressed as below.

Take any sample of n data points,

$$Z = \{(x_{11}, \ldots, x_{1p}, y_1), \ldots, (x_{n1}, \ldots, x_{np}, y_n)\}, \tag{3.10}$$

and let T be a estimation estimator. This means that applying T to Z yields a vector of estimation coefficients:

$$T(Z) = \hat{\boldsymbol{\theta}}. \tag{3.11}$$

Now consider all possible corrupted samples Z' that are obtained by replacing any m of the original data points by *arbitrary values* (this allows very bad outliers). Let us denote by bias$(m; T, Z)$ the maximum bias that can be produced by such contamination:

$$\text{bias}(m; T, Z) = \sup_{Z'} \|T(Z') - T(Z)\| \tag{3.12}$$

where the supremum is over all possible Z'. If bias$(m; T, Z)$ is infinite, this means that m outliers can have arbitrarily large effect on T, which may be expressed by saying that the estimator "breaks down". Therefore the (finite sample) *breakdown point* of the estimator T at sample Z is defined as

$$\epsilon_n^*(T, Z) = \min\{\frac{m}{n}; \text{bias}(m; T, Z) \text{ is arbitrary}\} \tag{3.13}$$

In other words, it is the smallest fraction of contamination that can cause the estimator T to take on values arbitrarily far away from $T(Z)$.

[1]Rousseeuw and Leroy [69] also use the same definitions.

Outliers normally constitute a very small portion of the total sample space, namely:

$$m \ll n. \tag{3.11}$$

However, for some applications the estimator must be highly robust even when outliers take a large proportion of the sample space. For example, in range image segmentation, a region may consist of several segments, each segment being one primitive model to be extracted. Therefore, if we concentrate on one segment, the others should be considered as *outliers* to the primitive we are considering. The number of outliers in this case is the sum of points in all other segments, and this number may be larger than 50% of the total number in the sample space.

A robust estimator is one which has ability to resist the effect of outliers. The breakdown point is a measure of the robustness of the estimator. The higher the breakdown point, the more robust the estimator. Traditional estimators, such as the least square estimator, L_1 estimator, *etc.*, are not robust because one outlier may cause the estimate arbitrarily far from the correct estimation. Robust estimation is still an active research area. Huber [11] proposed M-estimator as early as 1964 for estimation of the location and scale parameters from a sequence of independent and identically distributed (iid) observations. Other estimators, such as L-estimator and R-estimator [11], are similar to M-estimator. Recently, Zhuang, Wang and Zhang [89] proposed MF estimator. But all these estimators have breakdown points much smaller than 50%.

3.3 Linear Least Squares Approach

The linear least squares approach (LS) is the most popular estimation analysis method due to the following properties.

- it is an optimal estimator for data contaminated by Gaussian noise.

- it is highly efficient for its linear solution.

LS method can be expressed as follows:

1. Form matrix **M**:

$$\mathbf{M} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_m \end{bmatrix}. \tag{3.15}$$

   where $\mathbf{x}_i$'s are instantiated models with p terms. **M** is an $m \times p$ matrix.

2. The estimation equation is of the form.

$$\mathbf{M}\boldsymbol{\theta} = \mathbf{B}. \tag{3.16}$$

   where **B** is a $m \times 1$ column vector consisting of the instances of y in equation (3.1). and $\boldsymbol{\theta}$ is the parameter vector $p \times 1$.

3. For $m > p$, equation (3.16) is called an overdetermined system. It can be solved by *pseudo-inverse* method:

$$\hat{\boldsymbol{\theta}} = (\mathbf{M}\mathbf{M}')^{-1}\mathbf{M}'\mathbf{B} \tag{3.17}$$

   Some other methods, such as QR decomposition and eigensystem method [59, 52], could give better but more complicated solutions.

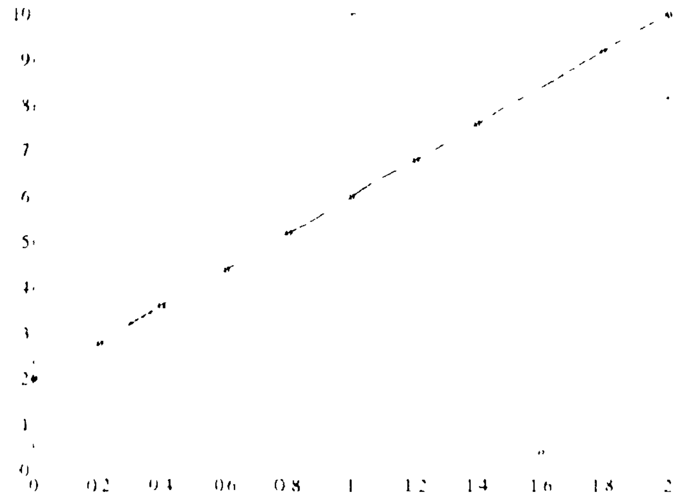
LS minimizes the sum of the squared residuals:

$$\min_{\boldsymbol{\theta}}(\mathbf{R} \cdot \mathbf{R}) = \min_{\boldsymbol{\theta}} \sum_{i=1}^{m} r_i^2 \tag{3.18}$$

where $\mathbf{R} = \mathbf{M}\boldsymbol{\theta} - \mathbf{B}$ and r_i is the element in **R**.

The traditional least squares algorithm has a breakdown point of 0%, because one outlier may cause the method to fail. Consider a simple situation where the data are generated from a straight line in a two dimensional coordinate system. More con

Figure 3.1: One outlier causes the failure of least squares fitting.



Note: (1) symbol 'o' represents a data point; (2) the dotted line is fitting by least squares method; (3) the solid line is fitting by RESC method.

point from the original position to a biased location in z-direction. This point is an outlier. This case is illustrated in Figure 3.1, as well as the fitting results by LS and RESC methods (explained in the next chapter). From the example we can see clearly that LS fails to find a best fit to this set of data. The example shows only the outlier biased in y-direction. If an outlier is biased in x-direction, the effect is even more drastic[2] than in y-direction because the minimization objective is to minimize the residuals in y direction as defined in equation (3.8). Since residuals in range images are in y direction, we do not give examples of outliers in x-direction.

For the bivariate polynomial fitting

$$z(m, a, x, y) = \sum_{i+j \le m} a_{ij} x^i y^j, \qquad (3.19)$$

[2] See examples in [69]

Figure 3.2: The geometry of fitting error around a conic section

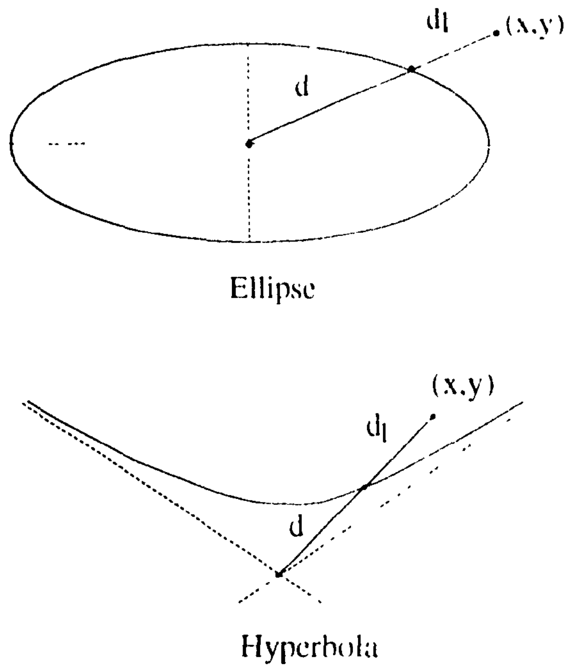
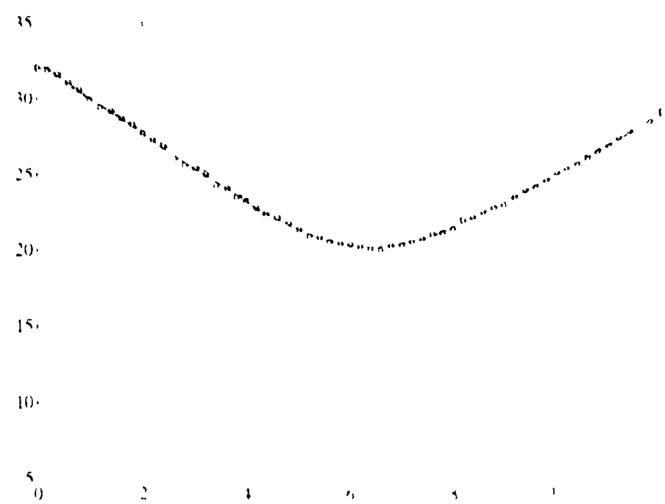


Figure 3.3: The least squares fitting of a conic curve



Note: (1) symbol 'o' represents a data point; (2) the dotted curve represents fitting by the least squares method; (3) the solid curve represents fitting by RESC method.

where usually $0 \leq m \leq 1$, the least squares method performs very well if a correct segmentation and outlier-free case is assumed. LS minimizes the geometric distance between the fitting surface and the actual z values. Consider a general quadratic form [22]:

$$Q(\mathbf{x}) = \mathbf{x}^t \mathbf{A} \mathbf{x} + \mathbf{x} \cdot \mathbf{v} + d \tag{3.20}$$

where

$$\mathbf{A} = \begin{bmatrix} a_1 & a_4/\sqrt{2} & a_5/\sqrt{2} \\ a_4/\sqrt{2} & a_2 & a_6/\sqrt{2} \\ a_5/\sqrt{2} & a_6/\sqrt{2} & a_3 \end{bmatrix}, \tag{3.21}$$

$$\mathbf{v} = (a_7 \ \ a_8 \ \ a_9)^t, \tag{3.22}$$

$$d = a_{10} \tag{3.23}$$

$$\mathbf{x} = (x \ \ y \ \ z). \tag{3.24}$$

The LS fitting minimizes:

$$E = \sum_{i=1}^{N} Q(\mathbf{x}_i)^2. \tag{3.25}$$

Various constraint methods are used, such as $d = 1$ in [37], $\mathrm{Tr}(\mathbf{A}\mathbf{A}^t) = \sum_{i=1}^{n} a_i^2 = 1$ in [22] to ensure invariance to geometric transformations. All these methods cannot express z, or y in 2D case, explicitly. But the noise contamination in the range image is mostly in z-direction, or y-direction in 2D range image profiles. The value of $Q(x, y)$ in equation (3.20) is proportional to $(d + d_1)^2/d^2 - 1$, as shown in Figure 3.2 [9] and it is simply called algebraic distance (the difference between the two sides of the equation) [62, 28, 29]. LS method minimizes algebraic distance $Q(x, y)$ instead of required geometric distance between the fitting surface and the actual data in z-direction. In this case, LS cannot even tolerate normally distributed Gaussian noise. Figure 3.3 shows the least squares fitting of a conic curve to the synthetic data with very low level Gaussian noise ($\sigma = 0.05$). It has to be noted that the least squares fitting failure in this case does not imply that LS cannot tolerate even normally distributed noise in the outlier free cases. The problem is that we cannot express the implicit equation in a way that the LS can effectively minimize the required geometric distances. Therefore, the problem is not with LS itself in the outlier free case, but with the way it is being used.

Taubin [78] derives approximate distance for implicit form of curves or surfaces. The approximate distance is in the direction perpendicular to the surface normal whereas the error of real range image in mainly in the z direction. The minimization of the approximate mean square distance is a nonlinear least squares problem. Although in certain cases, this problem reduces to the generalized eigenvector fit, in general cases, the iterative Levenberg-Marquardt algorithm has to be used. The computation is extensive for such iterative algorithm [78], therefore, in most cases, a simplified fitting is used in the segmentation algorithm.

3.4 Summary

In this chapter, we explained estimation models and the commonly used least square method. The outlier concept has also been introduced. Since the least square method cannot tolerate any outliers and cannot be effectively used in fitting the second-order surface primitives, other estimation methods should be explored. Taking into account the requirement of the second order primitive fitting, we propose in the next chapter, a new robust estimation method with a high breakdown point

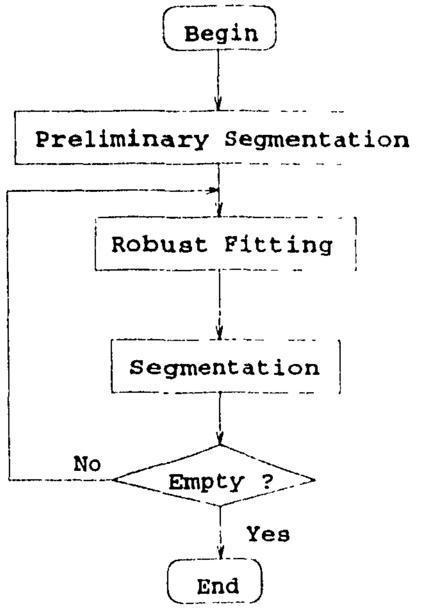
Chapter 4

Robust Estimation by Residual Consensus (RESC)

Our fitting and segmentation process is illustrated in Figure 1.1. A preliminary segmentation (jump edge detector) is applied to raw image. Each preliminary segmented region may contain several smooth connected regions. We use a robust estimation method to extract primitives from the regions. The process is repeated until the whole region segmented into primitives. The key issue is the robust estimation process.

In this chapter, we propose a robust estimation method which estimates primitive parameters of the largest homogeneous surface patch in the current processing region from noisy image data and then removes this patch from the processing region. A good surface fitting usually implies a good segmentation. In our method the fitting and segmentation are performed simultaneously. The method *randomly* samples p image points (p depends on the chosen fitting type of primitive, whether

Figure 4.1: Fitting and Segmentation



it is planar or quadratic) and solves the equation for the primitive parameter. From K samples we select the one having the best *residual consensus*, i.e. for which the most residual values are concentrated in a small range. To measure the consensus, we build a histogram of residuals and select a sample which has residuals concentrated within a small range in the lower part of the histogram. We introduce a concept of *histogram power* to represent this criterion quantitatively. An ordinary histogram works well only with a fixed noise level. We introduce a *compressed histogram* to measure and compare residuals at various noise levels. Outliers by definition are in the right part of the histogram and are neglected. Thus the selected data are homogeneous and their standard deviation σ can be calculated easily from the histogram. We find the maximum continuous region in which the residuals are within a tolerance range determined by σ. This region is the segment and its parameters are already determined.

The preliminary version of this algorithm was published in the *Proceedings of*

IEEE 1992 Computer Vision and Pattern Recognition [86], *Proceedings of the SPIE Advances in Intelligent Robotic Systems, Sensor Fusion IV: Control Paradigms and Data Structures* [83] and *Proceedings of the Canadian Conference on Electrical and Computer Engineering* [84]. It was also submitted to *IEEE Transactions on Pattern Analysis and Machine Intelligence* [87]

In Section 4.1, we explain the concepts of random sampling technique. The equations for the expected number of samples are derived. Random sampling has the advantage of outlier insensitivity. In Section 4.2, we propose our RESC algorithm which is based on random sampling principle and the compressed histogram technique to measure residual consensus at different noise levels. In order to further speed up the algorithm, we solve the large region problem by initial segmentation and lower resolution method, and we use a region mapping method to ensure an efficient and uniform sampling mechanism. In Chapter 6, we describe the segmentation algorithm in detail. In Section 4.3, we describe a method to switch different primitive types in a region. Instead of using variable-order surface fitting algorithm by Besl and Jain [7, 5] and Taubin [78], we use invariants extracted from surface parameters to determine the surface type, therefore avoiding repeated processing of two different primitives for every region. We compare our method with others in Section 4.4 and demonstrate experimental results in Chapter 7. Section 4.5 summaries the chapter.

4.1 Random Sampling

Parameters of a primitive surface can be determined by p points: a plane is determined by 3 points and a quadratic surface by 9 points. The key problem is how to choose these p points, using the data from a region, to determine a primitive surface which best fits the region. This is different from optimization approach which determines the parameters directly from all points in the processing region using certain optimization criterion such as the least squares method

4.1.1 Number of Combinations

The number of ways p points can be chosen from a sample space with n points is a huge number. Suppose that there are $256 \times 256 = 65536$ points in input data and we want to fit a quadratic surface to the data. A quadratic surface can be determined by 9 points. Therefore, the number of choices equals the number of combinations of 65536 points taken 9 at a time:

$$\binom{n}{p} = \binom{65536}{9} \approx 6 \cdot 10^{37} \tag{4.1}$$

Even for a surface patch with 200 pixels, this number can be as large as 10^{15}. Therefore, in practice a complete combinatorial search is impossible.

4.1.2 Principle of Random Sampling

To overcome this difficulty, random sampling method can be used. Random sampling methods have recently been widely used in computer vision research, for example in RANSAC[24], LMS[69, 66], CBD[58], RHT[80], etc. Random sampling is a process to select one element s from sample space S and every element in S has equal probability to be chosen. We denote such process by $\pi(S)$. For an instance of a model with p points, repeat the random sample process p times:

$$r_i = \pi(S), \quad \text{for } i = 1, \ldots, p \tag{4.2}$$

Assume the primitive in the sample space has m points, then the probability of sampling the primitive from a sample space with n points is

$$r = \frac{m}{n} \tag{4.3}$$

If we assume that there is only *one* sample set which is the best solution for the model, the random sampling does not help because the probability of finding such sample is very low.

$$r^* \approx \frac{1}{n} \tag{4.4}$$

For the example above, this probability can be as low as:

$$r^* \approx \frac{1}{65535^9} \approx 5 \times 10^{-44}, \tag{4.5}$$

so search complexity is even worse than deterministic combinatorial search (see Equation 4.1). But in practice, we do not have to obtain the best solution. An approximate best solution is normally good enough for practical applications. A good sample set means that all points in the set are on the primitive to be extracted and all these points have only small bias by the noise. The good samples may have more than one set. The combinatorial search must search every possibility before a solution can be found. Whereas random sampling method does not have to select all possible samples, the number of samplings depends on the fitting requirement. If the goodness measure of a sample is good enough, or the number of samples exceeds predefined limit (to keep down the cost of computation), we can stop the sampling process.

4.1.3 Expected Number of Sample Set

The following derivations gives an estimate of the number of trials needed to obtain a good sample [24]. Let r be the probability that a sample is a good one. Define $\alpha = r^p$ the probability that all p samples are good and $\beta = 1 - \alpha$ is the probability that all p sample points are bad. The expected number of samples K is:

$$\begin{aligned} E(K) &= \sum_{k=0}^{\infty} K \cdot prob(K) \\ &= \alpha + 2(1-\alpha)\alpha + 3(1-\alpha)^2\alpha + \cdots + K(1-\alpha)^{K-1}\alpha + \cdots \\ &= \alpha(1 + 2\beta + 3\beta^2 + \cdots + K\beta^{K-1} + \cdots). \end{aligned} \tag{4.6}$$

To express the above equation explicitly, consider an identity for the sum of geometric series:

$$\frac{x}{1-x} = x + x^2 + x^3 + \cdots + x^n + \cdots \tag{4.7}$$

Differentiating the above equation with respect to x, we have:

$$\frac{1}{(1-x)^2} = 1 + 2x + 3x^2 + \cdots + nx^{n-1} + \cdots \tag{4.8}$$

Table 4.1: Expected number of samples

r	$p = 1$	2	3	4	5	6	7	8	9
0.9	1.1	1.2	1.4	1.5	1.7	1.9	2.1	2.3	2.6
0.8	1.2	1.6	2.0	2.4	3.1	3.8	4.8	6.0	7.5
0.7	1.4	2.0	2.9	4.2	5.9	8.5	12	17	25
0.6	1.7	2.8	4.6	7.7	13	21	36	60	99
0.5	2.0	4.0	8.0	16	32	64	128	256	512
0.4	2.5	6.3	16	39	98	244	610	1526	3815
0.3	3.3	11	37	123	412	1372	4572	15241	50805
0.2	5.0	25	125	625	3125	15625	78125	390625	1953125

Upon replacing x by β, the Equation (4.6) can be rewritten as

$$E(K) = r^{-p} \tag{4.9}$$

A number of values for Equation (4.9) are listed in Table 4.1. From the table, we can see that the expected number is much smaller than the number required by a complete combinatorial search.

The standard deviation of K can be calculated as follows:

$$SD(K) = \sqrt{E(K^2) - E(K)^2} \tag{4.10}$$

Since

$$\begin{aligned} E(K^2) &= \sum_{i=0}^{\infty}(i^2\alpha\beta^{i-1}) \\ &= \sum_{i=0}^{\infty}[i(i-1)]\alpha\beta^{i-1} + \sum_{i=0}^{\infty}(i\alpha\beta^{i-1}) \end{aligned} \tag{4.11}$$

and since the second order derivative of Equation (4.7) is

$$\frac{2}{(1-r)^3} = \sum_{i=0}^{\infty}(i(i-1)\beta^{i-2}) \tag{4.12}$$

we have

$$E(K^2) = \frac{2-\alpha}{\alpha^2}. \tag{4.13}$$

The standard deviation of K is:

$$SD(K) = \sqrt{1-\alpha}/\alpha = r^{-p}\sqrt{1-r^p}. \tag{4.14}$$

Normally, $r^p \ll 1$, therefore,

$$SD(K) \approx r^{-p} = E(K). \tag{4.15}$$

Random sampling method greatly accelerates the search speed but still maintains the high probability of finding a good solution provided we have enough samples. All random sampling methods (RANSAC[21], LMS[69, 66, 68], CBD[58], RHT[80], *etc.*) are based on this principle.

4.1.4 Outlier Insensitivity

In addition to the reduction of combinatorial search, random sampling scheme also has the advantage of insensitivity to outliers, where an *outlier* is defined as point with a large residual (see section 3.2 for details). Outliers may not be just individual exceptions: another surface patch may be a set of outliers. The inliers are the points in the region excluding outliers. Therefore, our segmentation and fitting process finds the largest homogeneous region which is the expected primitive, and considers all others as outliers. Random sampling method will not fail when outliers exist. Outliers only reduce probability r. Suppose that the sample space has more than one primitive model, ith model has m_i points, $i = 1, \ldots, M$. Therefore, the probability that the sample point is on the ith model is:

$$r_i = \frac{m_i}{n}, \tag{4.16}$$

where $n = \sum_{i=1}^{M} m_i$. It is obvious that the more points in a model, the higher the probability of choosing one point in that model. Therefore, the random sampling

Table 4.2: Minimum number of sample sets for 99% assurance

r	$p=1$	2	3	4	5	6	7	8	9
0.9	2	3	4	4	5	6	7	8	9
0.8	3	5	6	9	12	15	20	25	32
0.7	4	7	11	17	25	37	54	78	112
0.6	5	10	19	33	57	96	162	272	455
0.5	7	16	34	71	145	292	587	1177	2356
0.4	9	26	70	178	447	1122	2808	7025	17565
0.3	13	49	168	566	1893	6315	21055	70188	233965
0.2	21	113	573	2876	14389	71953	359777	1798893	8994473

process will extract the largest model in the sample space and consider all other points as outliers.

Suppose that r is the percentage of inliers in a region. For a primitive model with p parameters, the probability of all p good sample points is r^p. The probability for all K set samples being outliers is $(1 - r^p)^K$. Therefore the probability of at least one of the set being a good one is:

$$\hat{\rho} = 1 - (1 - r^p)^K . \quad (4.17)$$

The minimum number of sample set K which contain at least one good point with probability $\hat{\rho}$ is given by:

$$K = \frac{\log(1 - \hat{\rho})}{\log(1 - r^p)} \quad (4.18)$$

For example, if $r = 50\%$ and $p = 3$, then $K = 34$ with 99% confidence. Table 4.2 shows the minimum number of sample sets containing at least one good selection at 99% confidence level.

The pure random sampling is still slow because the probability of finding a good sample is low when the number of points for a model is large or the number of good points in the sample space is small, as in case of very noisy data. A genetic algorithm can be used to accelerate search speed and maintain the advantage of random sampling. The genetic algorithm is explained in chapter 5.

4.2 Primitive Fitting by Residual Consensus (RESC) Method

4.2.1 The Algorithm

The method uses the random sampling technique and performs residual analysis for each sample set using an iterative algorithm, seeking a *RESidual Consensus* (RESC). The RESC estimator is highly robust with respect to outliers.

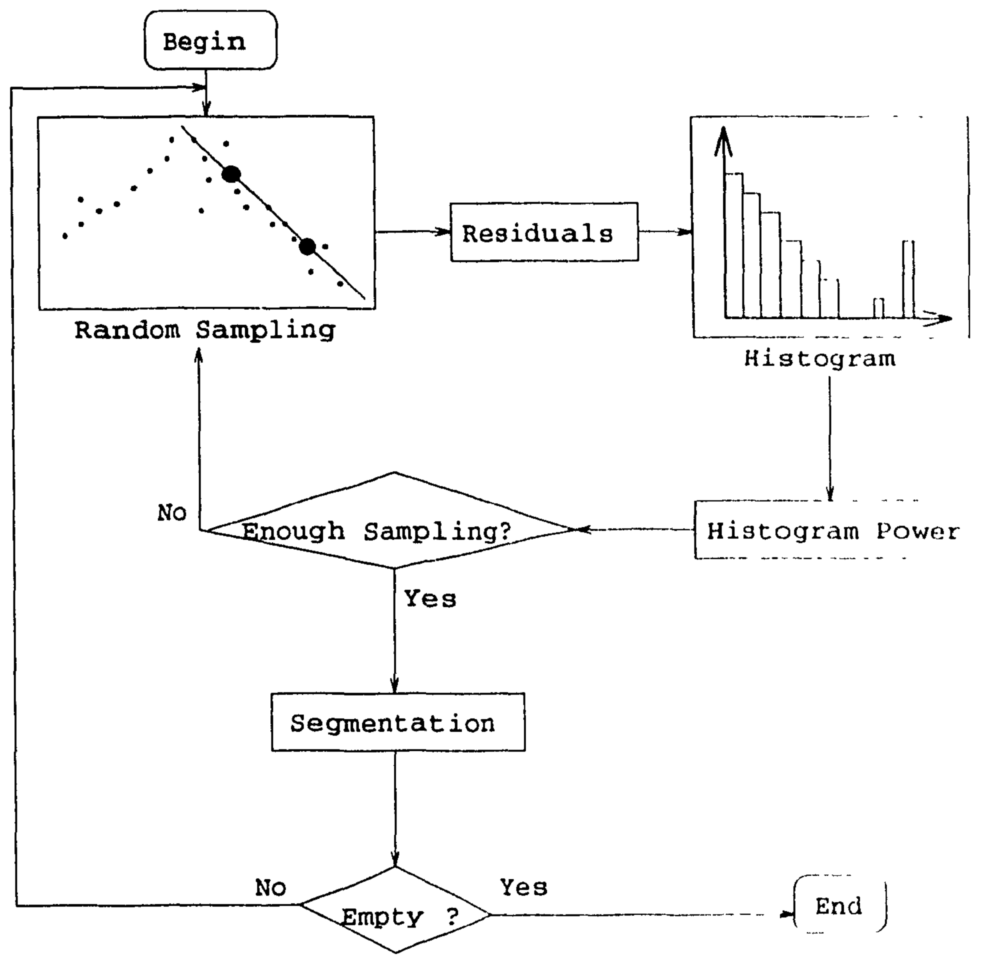
The algorithm finds in each iteration a parameter set $\boldsymbol{\theta}$ which is the solution of the equation $F(\mathbf{X}, \boldsymbol{\theta}) = 0$, where $\mathbf{X}$ is a vector of points. From K sample sets we find the largest continuous region where the residuals tend to be minimum. The residual at point i is defined as,

$$r_i = z_i - z_i'. \tag{4.19}$$

where z_i is z value of the range image at position i and z_i' is z value calculated from the fitted equation. Function F is the equation of the primitive. Note that for the linear primitive this definition is the same with that in Equation (3.8), but for the second order primitive, it is not the same. The definition here has clear geometric meaning and is consistent with real situations where noise influences the range image mainly in z direction.

The basic RESC algorithm concept is illustrated in Figure 4.2. The details is described in Figure 4.3. Although RESC can be used in various areas where robust estimation is needed, we focus our attention on the applications of RESC in range

Figure 4.2: REsidual Consensus (RESC) Algorithm



1. Randomly sample K sets of p points ($p = 3$ or 9) from the current sample space S.

2. For each set of points calculate the residuals of raw data using the primitive determined by these points, and make a histogram of the residuals.

3. From the K sets select the one whose histogram shows greatest *power* (explained below).

4. Determine from the histogram the standard deviation σ of the residuals, which is the noise level in the fitted surface region.

5. Label the points of this primitive and remove them from S, so that this set of data will not be included in further processing.

6. Remove outliers within the labeled region.

7. Repeat steps 1–6 until $S = \phi$.

Figure 4.3: The RESC algorithm

image segmentation and fitting.

It should be noticed that in the purely random search described here the number of sample sets K is typically large. Because except the outliers the range image is also contaminated by Gaussian noise, ratio r in Equation (4.3) is very small. This means that the expected number of samples is large. We adopt *genetic algorithm* (GA) instead of pure random search as proposed by Roth and Levine [67] to accelerate the search speed. The application of GA is explained in chapter 5.

4.2.2 Validation of a Sample Set

In step 1 of the RESC algorithm, each set of p points is validated before they are used to determine equation parameters.

A set of p points determines a matrix $\mathbf{X}$:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_p \end{bmatrix}, \tag{4.20}$$

where $\mathbf{x}$ is the model frame as defined in Equation (3.1). Its explicit form can be determined from Table 3.1 depending applications. A set of points is *valid* if the matrix $\mathbf{X}$ determined by these points is not singular or nearly singular:

$$\det(\mathbf{X}) > \epsilon \tag{4.21}$$

It is obvious that if there exits:

$$\mathbf{x}_i = \mathbf{x}_j, \quad \text{for } i \neq j \tag{4.22}$$

such that the rank of matrix $\mathbf{X}$ is lower than p, then $\mathbf{X}$ is singular. This means that if a point is repeated in the set, the set is invalid. In the random sample process this repeated set occurs with the probability $(1/n)^2$ for a sample space of n points

```
S = nil
for i := 1 to p step 1
    repeat
        r = random(1 : n);
    until r ∉ S
    put r into S
endfor
```

Figure 4.4: Random sample set generation and validation

In addition to condition in Equation (4.22), the matrix **X** in Equation (4.20) may still be singular if a second order primitive model is applied to a first order data set. The details of the derivation are provided in Appendix A.

Therefore, the condition for a valid sample is:

1. no repeated data points in a sample,
2. no data set corresponding to the low order primitive is used in the higher order primitive model.

The first condition is easy to check during the random number generation process as shown in Figure 4.4. Whenever we generate a random point, check if the point exists already in the sample set. If it exits, another point is generated. Note that the function `random` in the algorithm generates a random integer in the set $[1 \ldots n]$, where n is number of points in the sample space and S is the generated sample set.

The validation check of the second condition can only be performed on the matrix **X** by checking Equation (4.21). We check the determinant of the matrix during the solution process. Sander and Zucker [71] validate the fit by checking the condition number of the matrix. In our mathematical package, it is more convenient to check the determinant than the condition number. Suppose $\mathbf{X}$ is the matrix determined by the sample points. If $\det(\mathbf{X})$ is less than some threshold ϵ, then the matrix is considered singular. This criterion is also used to switch from the second order to the first-order primitive. When most of the samples are invalid using a second-order primitive equation in a processing region, the region is considered to be the first-order (see section 4.3).

If we find $\mathbf{X}$ is singular or nearly singular, we simply abandon this set of sample points and generate another set.

4.2.3 Compressed Histogram Technique

The compressed histogram technique is a key component in the RESC algorithm. It serves four major functions:

1. Separates the inlier part from outlier part in a region;
2. Measures the goodness of a fit;
3. Measures the noise level (standard deviation) of the inlier part;
4. Works at different noise levels.

A regular histogram can carry out the first three tasks. The compressed histogram method solves in addition the variable noise level problem.

Conventional Histogram

The histogram used in the algorithm consists of ordered "bins" of a fixed width δ, in which we accumulate the discretized residuals. It is constructed as follows.

$$h_i \leftarrow h_i + 1, \quad \text{if } (i-1)\delta \leq |r_j| < i\delta \quad \text{for} \quad j = 1, \ldots, n, \tag{1.23}$$

where n is the number of points in the current processing region. The value of column h_i represents the number of points whose absolute residuals $|r|$ satisfy $(i-1)\delta \leq |r| < i\delta$. The residuals greater than a given upper limit are discarded. To make the histogram work well, it is important to select the bin width δ properly. If δ is too large, all the residuals may accumulate in the first column, and if δ is too small, the distribution may be sparse or ragged, as shown in Figure 1.6 and Figure 1.8.

Compressed Histogram Algorithm

Different types of range sensors have different noise levels. Further, for a real range image, errors of a sensor vary according to the distance of the object from the sensor. Surfaces with different distances from the sensor may have different error levels. For the original histogram, selection of the interval δ depends on the noise level. To make the algorithm work for different noise levels, we use a *compressed-histogram* method, as listed in Figure 1.5. In the algorithm, the superscript c means compressed and h^c is the compressed histogram. First, δ is set to the smallest possible value, giving a large number of histogram columns H_c, say 2000. This ensures that in the small noise case the histogram can work properly. If the noise level is higher than the smallest one, the original histogram of the residuals may be *sparsely* distributed. The original histogram is irregular and distributed over a wide range if the noise level is high. We then compress it to a new histogram which better expresses the distribution of residuals. In the initialization stage of the algorithm, set a number to represent the minimum number of residuals in the first column.

[illegible] pm (1.24)

1. { initialization }

   $\delta \leftarrow \delta_{finest}, h_s \leftarrow pn, a \leftarrow 0;$

   $h'_i \leftarrow 0$, for $i \leftarrow 1, \ldots, max;$

2. { Determine the number of columns in the orignal histogram to be combined into one column in the compressed histogram }

   for $i \leftarrow 1$ **to** max **step** 1

   $\quad a \leftarrow a + h_i;$

   $\quad$ **if** $a \geq h_s$ **then** exit; { for loop }

   endfor

3. $h_1 \leftarrow a, \nu \leftarrow i, k \leftarrow 0;$

4. {Compress the remaining part of the histogram }

   for $i \leftarrow \nu + 1$ **to** max **step** 1

   $\quad h_k \leftarrow h_k + h_i;$

   $\quad$ **if** $i \bmod \nu = 0$ **then** $k \leftarrow k + 1$

   endfor

Figure 4.5: Histogram compression algorithm

Figure 4.6: Direct histogram 1

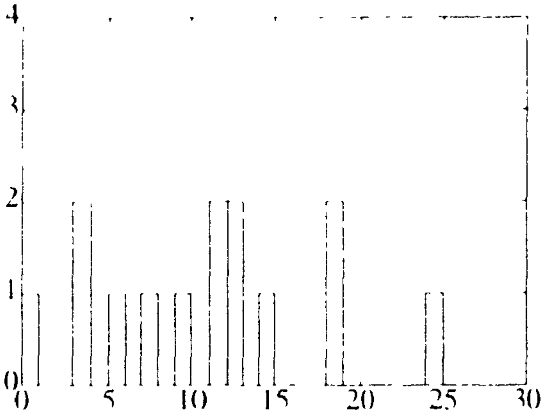


Figure 4.7: Compressed histogram 1

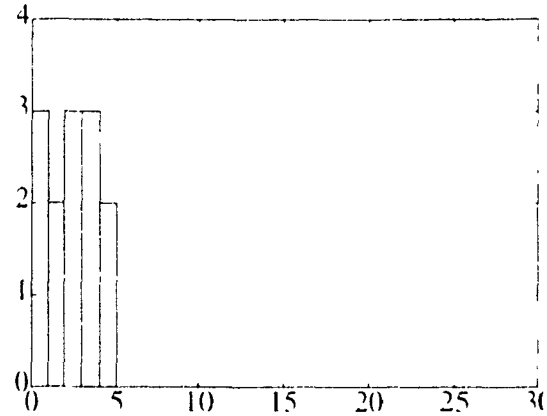


Figure 4.8: Direct histogram 2

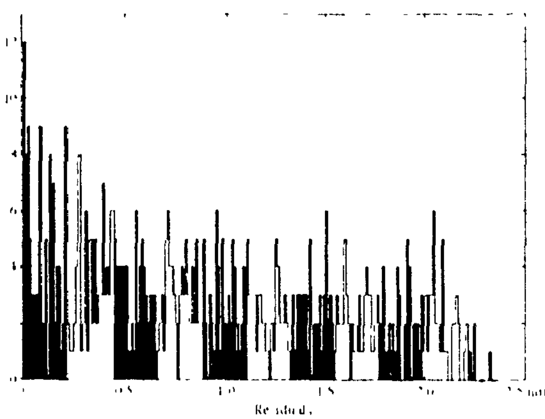
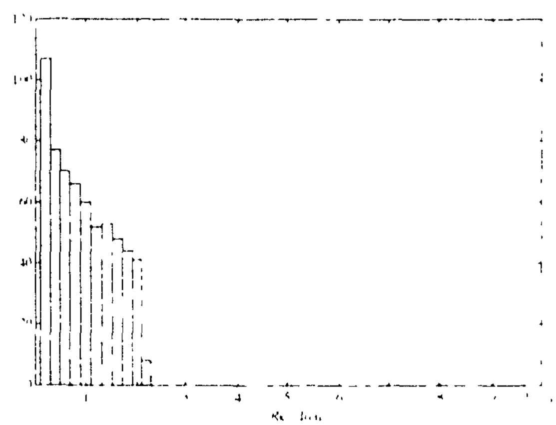


Figure 4.9: Compressed histogram 2



where n is the number of points in the current sample space and ρ is a coefficient. In step 2 of the algorithm, we determine the number of consecutive columns in the original histogram to be compressed into one column in the compressed histogram. In step 4, we compress every r columns into one in the compressed histogram.

Histogram Cutting Point

If the points are well chosen, i.e. on a primitive, the residuals on this primitive should be small and the histogram should be concentrated within a small range in the lower part of the histogram. We call this phenomenon the *residual consensus*

Points outside the range of concentration are considered to be outliers. We need to find the range of concentration, which represents the "good" data ("inliers"). We assume that the "good" residuals are described by Gaussian distribution

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}. \qquad (4.25)$$

where σ is the standard deviation and μ is the mean. Outliers are considered as all points with residuals larger than $\gamma\sigma$, where γ is a coefficient. Assuming $\mu = 0$, we compute the ratio:

$$f(\gamma\sigma)/f(0) = e^{-\gamma^2/2} \qquad (4.26)$$

This is the ratio of the maximum to the minimum probability of residuals. We usually set γ to 2.5 (see [69]), giving $f(\gamma\sigma)/f(0) \approx 4.4\%$.

The maximum value h_{max} of the histogram is set to be the number of residuals in the first column of the compressed histogram. We find the least i such that h_i is less than 4.4% of h_{max}, and stop further compression there. Any pixel in the current region with residual larger than this value is considered to be an outlier and is discarded. This is the point dividing inlier part from outlier part. Therefore the histogram compression solves not only different noise level problem, but also the outlier detection problem. Residuals in the compressed histogram are considered as inliers. The required primitive can be extracted from the points voted in the compressed histogram. Figure 4.6 shows a sparse distributed histogram which is obtained directly from the residuals of a set of sample points in a 2D profile. Figure 4.7 shows the compressed histogram from Figure 4.6, where $h_{max} = 3$ and $n = 4$. The last column in Figure 4.7 is discarded because $h_6 = 0 < ph_{max} = 0.044 \times 3 = 0.132$. When processing a region with large number of points, the directly built histogram is normally not sparsely distributed, but highly ragged as shown in Figure 4.8. It is not easy to find a rule which can distinguish the inlier and outlier parts. A compressed histogram makes this easy to determine. Figure 4.8 shows an original histogram. Consecutive filled columns or consecutive unfilled columns map to one column in the compressed histogram in Figure 4.9.

Histogram Power

The compressed histogram has a variable bin width. This makes it hard to compare histograms directly. For the purpose of optimization, we must determine an objective function based on the histogram analysis. There are two possible criteria, both of them should ideally be satisfied:

1. The number of points on and near the primitive surface should be as large as possible.

2. The residuals of the total inlier points should be as small as possible.

Several algorithms (e.g., RANSAC, RHT) use the first criterion as the objective. They count the number of points within an error band centered at the primitive. This number is the score of the optimization process. Methods using the second criterion are more commonly used for non-robust estimation. The most popular of these is the *least squares* method (L_2), which minimizes the sum of the squares of the residuals:

$$\min \sum_{i=1}^{n} r_i^2. \tag{1.27}$$

It is an optimal solution for residuals with a Gaussian distribution. The other popular criterion is L_1 method, which minimizes the summation of the absolute value of residuals:

$$\min \sum_{i=1}^{n} |r_i| \tag{1.28}$$

Since for larger residuals $|r|$ contributes less than r^2, L_1 is slightly better than L_2 when there exist a few outliers.

In our objective function, we combine the two criteria. For each column i of the histogram, we consider two factors,

1. h_i: the number of points in column i (Criterion 1), and

2. r_i, the residual of the column i (Criterion 2)

Our objective function is,

$$e = \sum_{i=1}^{m} \frac{h_i^{\alpha}}{|r_i|^{\beta}}, \qquad (4.29)$$

where m is the total number of columns in the compressed histogram and α and β are coefficients which determine the relative importance of the two factors. Considering only one of the factors is not enough for robust estimation. We cannot use the least squares criterion for the histogram analysis. A planar surface which is nearly normal to the actual surface will get the highest score because only a few points will be inliers and $\sum_{i=1}^{m} r_i^2$ will be very small. RANSAC [24], RHT [80] or other random sampling method [68] count only number of points in the error band. It works well on two dimensional images where the values are restricted on 256 × 256 or 512 × 512 grid depending on a correct selection of the width of the error band. In the range image data from NRC, the z values are floating point representation. It is inaccurate to count only the number of points in the histogram, because many different cases may have the same number of points.

The residual for column i can be expressed as

$$|r_i| = i\delta. \qquad (4.30)$$

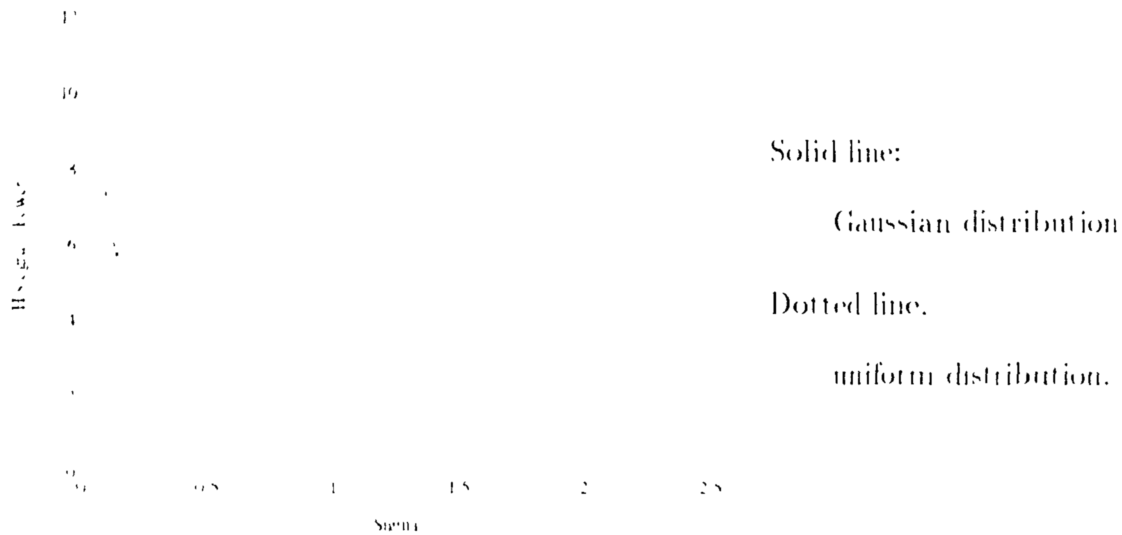
Since δ is a constant, it can be removed from the objective function. Therefore, our final objective function is:

$$e = \frac{1}{\delta^{\beta}} \sum_{i=1}^{m} h_i^{\alpha} i^{-\beta} \qquad (4.31)$$

The compressed histogram can be well described by a physical analogy. If we call h_i, the number of points accumulated in each column, the *work* which contributes to the primitive, and call the column index i the *time* representing the cost for the work done, then the objective function e is called the *power*. We use the empirically determined values $\alpha = 1.3$, $\beta = 1.0$.

The histogram power is monotonically decreasing with respect to the standard deviation σ, regardless of the probability distribution of the residual. [illegible]

Figure 4.10: Histogram power of Gaussian and uniform distribution



shows this property. This property of the histogram power ensures that the RFSC will choose those samples whose residuals are more concentrated. In Figure 4.10, the power e is given by

$$e = \sum_{i=1}^{50} f^2(i\delta)/i \; , \tag{4.32}$$

where f is the Gaussian density function (4.25) or the uniform distribution function, and δ, the histogram interval, is set to 0.05. The uniform distribution is:

$$f(x) = \begin{cases} \frac{1}{b-a}, & a < x < b \\ 0, & \text{otherwise} \end{cases} \tag{4.33}$$

Its standard deviation is:

$$SD(x) = \sqrt{\frac{(b-a)^2}{12}}. \tag{4.34}$$

In Figure 4.10, we plot the histogram power e against the standard deviation σ

Noise Level Estimation

After filtering out the outliers, the remaining residuals normally satisfy Gaussian distribution. The standard deviation σ of the best fitting can be calculated directly from the histogram:

$$\sigma = \xi\sqrt{\frac{1}{\sum_{i=1}^{n} h_i - 1}\sum_{i=1}^{n} h_i (i\delta - \bar{h})^2} \qquad (4.35)$$

where $\bar{h}$ is the mean of all residuals r included in the compressed histogram. The coefficient ξ corrects for the fact that in each column of the histogram all residuals are rounded to $i\delta$, which is greater than the actual values of the residuals in the column. Empirically we set $\xi = 0.88$.

4.2.4 Region Implementation

Regions are expressed in two dimensional array M by labels. Each region has a unique label. After the initial segmentation, each region is assigned a temporary label. In order to obtain easily a uniformly sampled point from a region, we use another one dimensional array R to map a region into M. The contents of R are indices of M. Therefore, for each region, one continuous range in R maps a region to pixels in M. Random sample points is drawn from R. A pointer to R is used to indicate the beginning of the region and another variable to indicate the number of pixels in the region. All regions are then represented by a linked list. After each segmentation, R has to be reorganized to maintain correct region mapping.

4.2.5 Large Region Problem

The RESC algorithm needs to calculate residual of each point in the processing region. The more points in the region, the more time is needed for each iteration. In practice the number of points in a region can be as large as 65536 (256 × 256) at the initial stage. Direct application of the RESC algorithm is too slow.

To speed up the algorithm, we make a preliminary segmentation [illegible] a simple jump edge detector to classify the whole range image into [illegible] regions. [illegible] details of the method are given in section 6.2. Each initial region would normally be much smaller than the whole original image.

Suppose that original region is R and it consists of m subregions $r_1, r_2, \ldots, r_m$, which can be separated by jump-edges. Without loss generality, we assume the size of each subregion is equal to s and the size of original region is $S = ms$. Without preliminary segmentation, RESC has to process initially S pixels, separating s pixels from R. It then processes $S - s$ pixels to get the second segmentation. The total number of pixels processed by the RESC method is:

$$
\begin{aligned}
N &= \sum_{i=1}^{m} [S - (i-1)s] \\
&= sm\frac{m+1}{2}. \qquad (4.36)
\end{aligned}
$$

With preliminary segmentation, the total pixels processed by the RESC method is:

$$N_p = sm \qquad (4.37)$$

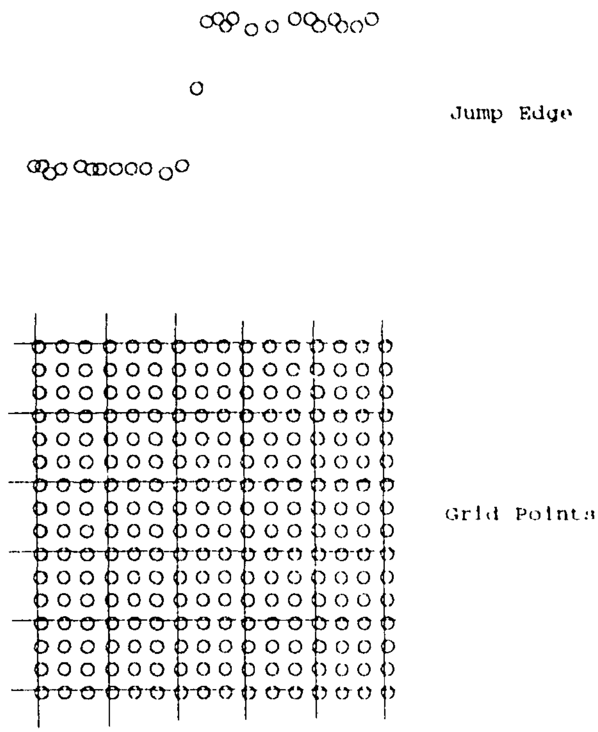
It is obvious that without preliminary segmentation, RESC has to process $(m+1)/2$ times more pixels than with the segmentation. Since the preliminary segmentation is much faster than RESC processing, we can save the total processing time by this strategy.

Here, the segmentation is only preliminary based solely on jump edges. Surface fitting is not performed at this stage. The more subtle edges or smoothly connected regions are then segmented and fitted by the RESC method.

Even these initial regions are sometimes too large to be processed fast enough. A simple solution to this problem is to lower the resolution of the range image of a large region temporarily during the fitting process. The number of pixels in a given region may be restricted to $N_{max} = 1000$ during the fitting process by sampling pixels only on a grid with spacing of k pixels, where k is determined by the size of the processing region:

$$k = \sqrt{\frac{n}{N_{max}}} + 1. \qquad (4.38)$$

Figure 4.11: Solve large region problem



where n_r is the number of points in the current processing region and N_{max} is the maximum number restricted for a region. This is shown in Figure 4.11. In experiments k varies from 1 to 6. The sampling points are chosen on the grid by the condition:

$$x \bmod k = 0 \quad \text{and} \quad y \bmod k = 0 \qquad (4.39)$$

Since a large region contains large homogeneous surface patches, lower resolution does not influence the accuracy of detection of large surface patches. In the segmentation phase, we have to use all the pixels in the region in order to get accurate segmentation.

4.3 Switching between Primitive Surface Types

We wish to segment range image into the first- and second-order primitives. The method described above handles each primitive type separately. We have to find a method to switch from one to the other.

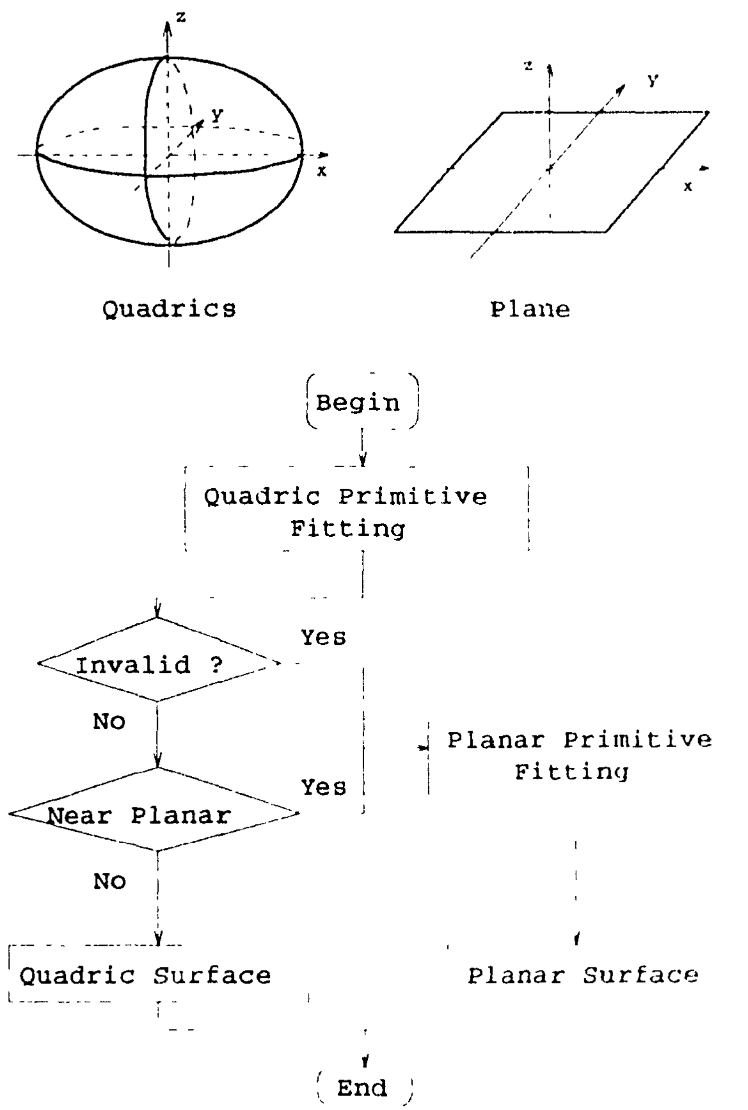
There are two possible strategies. One is to use a variable-order surface fitting algorithm [7, 5, 78]. If a curved region is approximated by the first order primitives, the region will be segmented into many small planar patches in order to get an accurate fitting. If a second order primitive is used to fit the curved region, the patches will normally be larger than the patches with the first order primitive fitting. By increasing the order of the primitive and comparing the number of pixels for different order primitive fitting, a suitable order can then be determined when there is no difference between the two orders. In this way, however, each region has to be fitted many times (at least twice) until a stable segmentation is obtained.

Using the other strategy, we fit the second order primitive first and then determine if it is truly second order from the properties of the fitted primitive. During the second order fitting, we check if the region is the first-order based on three factors:

1. validation detection,
2. invariant theory, and
3. average curvature.

If the surface is determined to be planar, we refit the region using the first order primitive. A flowchart is shown in Figure 4.12. We will explain each factor in the following.

Figure 4.12: Different primitive type switching



4.3.1 Switching by Validation Detection

For the first factor, we have mentioned before in subsection 4.2.2 that if a region is best described by a first order primitive with a very low noise level (e.g., synthetic data), the matrix for the second-order primitive will be singular or nearly singular (see Appendix A) which will also be indicated by its determinant. It may occasionally happen that *sample points on the second order surface degenerate* into first primitive. It is the case when all the sample points are on the intersection line (circle) of a sphere with a plane. But it is impossible for a random sample procedure to generate these special configuration very frequently.

The second order primitive is fitted to a region first. We monitor the number of invalid samples. If the rate of invalid sample to the total number of samples is greater than 10%, we assume that a first-order primitive is the best. We refit the current region with the first order primitive.

For most practical range images, there is enough noise in them so that it is unlikely that a second order equation determined by points in the region will have a singular matrix, even if the region is planar. In this case we check the quadratic invariants to determine if the region is the first order region.

4.3.2 Switching by Quadratic Invariant

The invariant theory of second order primitives is explained in detail in Appendix C. The surface type can be determined from these invariants as in Appendix D. Assume that invariants are (from Equation (C.81)):

$$\mathbf{I} = \{V_1', V_2', V_3', d'\}. \tag{4.10}$$

If the condition

$$|V_2'| < t \quad \text{and} \quad |V_3'| < t \tag{4.11}$$

holds, where T_p is a threshold, then the surface can be classified as a planar surface. Otherwise it is a second order primitive. In practice, we set $T_p = 10^{[illegible]}$.

4.3.3 Switching by Average Curvature

Another method to determine the surface type is the curvature method. The Gaussian and mean curvatures can be used to determine the surface type as shown in Table 2.1 in chapter 2. Gaussian and mean curvatures at one point of quadratic surface can be calculated by the formula in Appendix B. Average Gaussian and mean curvatures are:

$$\bar{K} = \frac{1}{n}\sum_{i=1}^{n} K_i \qquad (4.12)$$

$$\bar{H} = \frac{1}{n}\sum_{i=1}^{n} H_i \qquad (4.13)$$

where the summation is for n pixels in the current processing region. If the condition

$$|\bar{K}| < T_K \quad \text{and} \quad |\bar{H}| < T_H \qquad (4.14)$$

is true, the surface can be classified as a planar surface. In practice, we set $T_K = 10^{[illegible]}$ and $T_H = 10^{[illegible]}$.

We use all these three methods to switch primitive order. The second order primitive is used first, and if any of the methods described above indicates a lower order is desired, then the switch to the first order is carried out.

4.4 Comparisons with Other Methods

The most commonly used fitting method is the least squares method [illegible] criterion: $\min \sum_{i=1}^{n} r_i^2$ where r_i is the residual at point i in the first order [illegible] cases. It is widely used because of its elegant linear solution [illegible] the method has a breakdown point of 0%, therefore it cannot be used when outliers exist. Furthermore, for the second order primitive fitting, the least squares method is very sensitive to noise because of the implicit nature of the primitive equation [85, 82]. Residual r in this case is no longer the geometric residual. It is just the difference between the two sides of the fitting equation, called algebraic distance. Least squares method minimizes only this algebraic distance instead of the required geometric residual. Taubin [78] derives approximate distance for implicit form of curves or surfaces. The approximate distance is in the direction perpendicular to the surface normal, whereas the error of real range image is mainly in the z direction. The minimization of the *approximate mean square distance* is a nonlinear least squares problem. Although in certain cases, this problem reduces to the generalized eigenvector fit, in general cases, the iterative Levenberg-Marquardt algorithm has to be used. The computation is extensive for such iterative algorithm; therefore, in most cases, a simplified fitting is used in the segmentation algorithm. The method may fail in cases of outliers.

The RESC method uses a new criterion — residual consensus, i.e., finding the fitting which has residuals concentrated in the lower part of the histogram. We combine the two factors of the optimization as our objective function. *One* is the *number* of points in the primitive, another is the total deviation of the points. Therefore the residual consensus describes the case when both conditions are best satisfied. In case of Gaussian noise, the histogram will show Gaussian distribution. This means the solution is approximately optimal. The histogram method also has an advantage over the LMS criterion of selecting only the median of residuals because the median of squared errors can hardly represent the total inlier part. If the inliers are exactly 50% of the total points, the LMS *method uses the largest residual of the inliers*; if inliers are less than 50%, LMS fails. If there are no outliers in the region, the criterion of the median residual is a weak condition. This is proved in the experiment (see Chapter 7). A more flexible way is to use the histogram to determine the concentration of the error distribution. Whatever percentage of the inliers in the current region the histogram always shows the consensus of residuals and the inlier part is also the largest segment in current sample region.

Compressed histogram method can work at different noise levels. It does not depend on pre-obtained knowledge about the scene. Therefore it is better than RANSAC method.

From histogram method, inlier and outlier parts are easily determined from the histogram directly. The standard deviation of the fit can be estimated from the inlier part.

The time complexity of producing histogram is $\mathcal{O}(n)$, where n is the number of points to be considered. It is better than the LMS's sorting $\mathcal{O}(n \log_2 n)$.

Rousseeuw and Leroy [69] prove a theorem that the 50% breakdown point is the best a robust estimation method can achieve. But our experiments have demonstrated that RESC method has a breakdown point more than 80% (Figure 7.5, 7.13). In our cases, we do not consider the uniqueness of the fit. More analysis of the experimental results can be found in Chapter 7.

4.5 Summary

This chapter described the RESC method in detail. RESC method is based on the *random sample* principles. The essential part of the RESC algorithm is the *histogram method* for residual analyses. A compressed histogram method works on different noise levels. From the histogram, we can determine the *cutting point*, which separate inliers from outliers, the *histogram power*, which is the object function to be maximized, and the *standard deviation* of the noise for the inlier part. The RESC method is *highly robust* with respect to outliers, giving the breakdown point more than 80%. The RESC method is applied to range image segmentation and fitting.

By extracting one primitive at each time, the whole range image can be segmented into these primitives. The complete experiments for both synthetic and real data can be found in Chapter 7. The *genetic algorithm* can be incorporated into the RESC. This is explained in the next chapter.

Chapter 5

Genetic Algorithm

The RESC algorithm is based on random sampling of range image points to obtain the best fit of a primitive to a homogeneous surface patch. Pure random search normally takes a long time, however. A genetic algorithm (GA) can be used in step 1 of the RESC algorithm (Figure 4.3) to accelerate the search and to achieve the global optimal result.

The main results of this chapter will be published in the *Proceedings of the 8th Scandinavian Conference on Image Analysis* [88].

In section 5.1 we give a simple review of genetic algorithm. Section 5.2 introduces the basic concepts of GA. The terminology used in GA comes from biological adaptation system. In section 5.3, various GA operators are introduced. In section 5.4 we explain our GA used in the experiment. Instead of traditional generational replacement, we retain only one population (steady state system). In section 5.5, we explained what our gene is in order to incorporate GA to RESC algorithm. In section 5.6, experimental results for various GAs with variable control parameters are presented. It is proved that the steady state GA is better than the generational replacement GA if parameters are properly set. The best control parameter settings are quite different from that suggested by other researchers. The results are analysed. Section 5.7 summaries the chapter.

5.1 Review of Genetic Algorithm

Genetic algorithms have been developed by John Holland [11], his colleagues, and his students at the University of Michigan. Genetic algorithms are a class of optimization techniques that gain their name from a similarity to certain processes that occur at the interactions of biological genes. Basically, a genetic algorithm selects high strength *parent* models, forming *offspring* by recombining components from the parent models. The offspring replace weaker models in the system and enter into further competitions. Genetic algorithms have been studied intensively by Holland [11], Goldberg [31] and others [2, 36, 3, 72, 77, 73, 53]. Genetic algorithms have been widely used in various areas, such as: image processing and pattern recognition [25, 20, 31, 75], computer science [30, 35, 61, 63], engineering and operations research [33, 32, 12, 13]. The preliminary convergence properties are discussed in [54].

Roth and Levine [67] used GA in extracting primitives from 2D images. Hill and Taylor [39] also used GA in model based image interpretation. We used it here in 3D range image processing. In our GA, a non binary representation is used. Each gene is an index of the point in the current processing region and the value of the index is in the range of 1 to 1024. In this chapter, we study GA performance in this special situation and examine the influence of different parameter settings of GA on the RESC performance.

Since genetic algorithms are stochastic, the same parameter settings used for the same problem by the same genetic algorithm generally yield different results. To in case of noisy data. Several researchers did extensive experiments to determine the parameter settings [16, 36, 73, 11]. All their GAs solve optimization problem with the binary coding. This means that their genes consist of only two alleles, 0 and 1. Our gene is quite different. It consists of hundreds of different alleles. It is not clear that the parameter settings and GA performance are still the same. We tested two different GAs with various parameter settings. GA performance is illustrated in this chapter.

5.2 Basic Concepts

The idea and concepts of genetic algorithm comes from biological adaptation system. Most technical terms are inherited directly from biological system. Holland defines such terminologies in his book [11]:

> Every organism is an amalgam of characteristics determined by the *genes* in its *chromosomes*. A gene has several forms or alternatives — *alleles* — producing differences in the set of characteristics associated with that gene. (E.g., certain strains of garden pea have a single gene which determines blossom color, one allele causing the blossom to be white, the other pink; bread mold has a gene which in normal form causes synthesis of vitamin B_1, but several mutant alleles of the gene are deficient in this ability; human sickle cell anemia results from an abnormal allele of one of the genes determining the structure of hemoglobin — interestingly enough, in environments where malaria is endemic, the abnormal allele can confer an advantage.) There are tens of thousands of genes in the chromosomes of a typical vertebrate, each of which has several alleles.

If we translate these into our computer terminology, we can bring GA close to those who are not familiar with biological sciences. A *chromosome* is a *string* with p

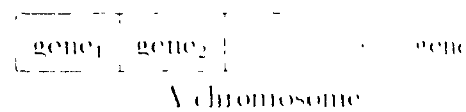


Figure 5.1: Chromosome and genes

elements in it. A *gene* is an element. An *allele* is an instance for a gene and its value is in the definition domain of the gene, see Figure 5.1. If we represent numbers in binary form, then an allele can take value 0 or 1; a chromosome with 5 genes may take the form

01101

The *performance* of a chromosome in the environment can be measured quantitatively. A measure of the performance is denoted by μ and called the *fitness*. All attainable chromosomes form a set $\mathcal{A}$. A subset of n chromosomes in $\mathcal{A}$ constitutes a *population* U. The initial population is normally created at random. A genetic algorithm is the process in which a set of genetic operators is applied to population U and a global optimal solution can be achieved.

5.3 Genetic Operators

The major three genetic operators are

1. reproduction,
2. cross over, and
3. mutation.

A simple genetic algorithm may normally yield good results in many applications with the three operators. Reproduction is a process to select appropriate chromosomes from the population according to some rules. The selected chromosomes, called parents P are then subject to the cross over and mutation operators to generate new chromosomes, called children C. The measure μ of the fitness of C is computed and if C is strong enough, it will replace the weakest chromosome in the population. The process continues until the stopping criterion is satisfied.

Reproduction operator selects one chromosome from population I probabilistically after assigning each chromosome a probability proportional to its observed performance. Intuitively, we can think that performance μ is the objective function we want to maximize. Selecting chromosomes according to their performance means that chromosomes with higher performance have higher probability of contributing one or more offspring in the next generation.

The reproduction operator can be implemented by a biased roulette wheel method [34] where each chromosome in the population has a roulette wheel slot of the size proportional to its fitness. A random draw from the population is equivalent to the rolling of the roulette wheel. The final stop position is the selection. Since the slot size is proportional to the fitness, the selection is also proportional to the fitness

$$probability(c_i) = \frac{score(c_i)}{\sum_{i=1}^{n} score(c_i)} \tag{5.1}$$

where c_i is the ith chromosome in the population and $score(c_i)$ is its fitness measure.

Another implementation of the reproduction is by ranking the population by performance. The random draw is probabilistically proportional to the ranking.

$$probability(c_i) = \frac{rank(c_i)}{\sum_{i=1}^{n} rank(c_i)} \tag{5.2}$$

The advantage of the ranking is in preventing some extremely strong members of the population to dominate the selection and causing premature convergence.

Cross over operator is described in Figure 5.2. Cross-over operator randomly selects position i, $1 \leq i \leq p$, where p is the minimum number of points for a given

1. Given two parents: $P^1 = P^1_1 P^1_2 \ldots P^1_p$ and $P^2 = P^2_1 P^2_2 \ldots P^2_p$

2. A number x is selected from $\{1, 2, \ldots, p-1\}$ at random.

3. Two new chromosomes are formed from P^1 and P^2 by exchanging the set of genes to the right of position x, yielding

$$C^1 = P^1_1, \ldots, P^1_x P^2_{x+1} \ldots P^2_p \quad (5.3)$$

$$C^2 = P^2_1, \ldots, P^2_x P^1_{x+1} \ldots P^1_p \quad (5.4)$$

where C^1 and C^2 are children of their parents P^1 and P^2.

Figure 5.2: A simple crossover operator

primitive type. From two parents, the cross over operator exchanges all points at x and thereafter. For example, assume that parent structures for a quadratic surface primitive are :

$$a_1, \; a_2, \; a_3, \; a_4, \; a_5, \; a_6, \; a_7, \; a_8, \; a_9$$
$$b_1, \; b_2, \; b_3, \; b_4, \; b_5, \; b_6, \; b_7, \; b_8, \; b_9$$

Suppose $x = 7$ by a random selection. After cross over operation, the two offsprings are :

$$a_1, \; a_2, \; a_3, \; a_4, \; a_5, \; a_6, \; b_7, \; b_8, \; b_9$$
$$b_1, \; b_2, \; b_3, \; b_4, \; b_5, \; b_6, \; a_7, \; a_8, \; a_9$$

Besides a simple crossover operator developed by Holland, other crossover operators were introduced. Uniform crossover operator [77] is widely used. It is described in Figure 5.3.

Mutation operator is normally applied after crossover operator. When crossover operator generates new chromosomes, a mutation operator is applied to each gene. The mutation operator replaces, with a given probability, gene x in parent [illegible] an allele randomly selected from the domain of gene. A mutation operator [illegible]

1. Given two parents $P^1 = P^1_1 P^1_2 \ldots P^1_p$ and $P^2 = P^2_1 P^2_2 \ldots P^2_p$;

2. R is a set with numbers from 1 to p and $g = p/2$;

3. { Randomly copy half genes from parents }

for $i = 1$ **to** g **step** 1
    r = a random selection from R;
    copy P^1_r to C^1_r;
    copy P^2_r to C^2_r;
    $R = R - r$;
endfor

4. { copy another half genes from parents }

while $R \neq$ **nil**
    take an element r from R;
    copy P^1_r to C^2_r;
    copy P^2_r to C^1_r;
    $R = R - r$;
endfor

Figure 5.3: Uniform crossover operator

```
if probability(mutation_rate) = true
then
   new_gene_i = random selected point from R
else
   new_gene_i = old_gene_i.
endif
```

Figure 5.4: Mutation operator

in Figure 5.3.

5.4 Genetic Algorithm

Most genetic algorithm structures are of the generational replacement type [11, 16, 34]. We call this algorithm GA1 as shown in Figure 5.5. A reproduction operator selects parents P_1 and P_2 from $P(t)$. Crossover is applied to P_i with a given probability X_c to generate children C_1 and C_2. Mutation is applied to C_i with a given probability M, and a new generation $P(t+1)$ is formed. Note that if P_i are not performed crossover operation, it is copied to $P(t+1)$ directly.

In this thesis another approach called GA2 is used. Only one population is maintained. Parents are selected from P by reproduction operator. They are subjected to crossover operator followed by mutation operator. Children C_i generated by genetic operators are evaluated and inserted in P according to their fitness. The worst chromosomes are then discarded from the population forthwith. This is a steady state system [11, 79, 15]. GA1 and GA2 are tested in various parameter settings as explained in the section on GA experiment. It is proved that GA2 has a

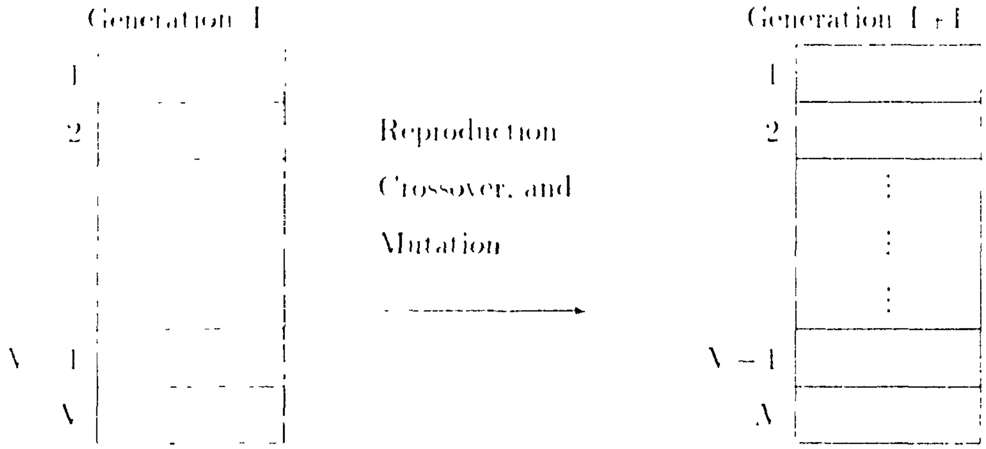


Figure 5.5. Schematic of non-overlapping population generations

faster search speed than GA1.

Initial population is created by random sampling method until the size of the population U reaches a given level. The point set **X** stored in U is sorted according to its performance, i.e., the histogram power. The genetic algorithm used in the experiment is described in Figure 5.6.

In our GA algorithm, we do not allow repeated chromosomes in the population due to the following reasons:

1. the redundant chromosome in the population will have much higher probability than other chromosomes. For the same population size, the number of different chromosomes is reduced by such redundancy. It limits the global search ability of GA and may cause convergence (all chromosomes are the same in the population) to a local optimum.
2. redundant chromosome results in redundant evaluation of the chromosome. It wastes time to calculate the results which are already known.
3. checking chromosome redundancy in the population costs less than evaluation of the chromosome.

1. Select and copy two chromosomes from current population P probabilistically proportionally to their ranking (all chromosomes are ranked by its performance, i.e., histogram power)

2. Apply cross-over operator to the selected chromosomes and generate two new chromosomes.

3. Apply mutation operator to each gene of the new chromosome with a given probability

4. Check if the new chromosome exists in P already

   (a) If it is the case, abandon this chromosome

      i. If all new chromosomes are checked, goto step 1

      ii. Otherwise, goto step 4 to check another chromosome

   (b) Otherwise goto step 5

5. Evaluate the performance of the new chromosome(s)

6. Insert the new chromosome(s) into population P according to their performance. This may cause some chromosomes with lowest performance to be eliminated from current population

7. Repeat the above steps until the difference of the highest performance and the second highest performance is less than a pre-defined constant or the number of offspring reaches a given level

Figure 5.6: The genetic algorithm

Since there are no redundant chromosomes in the population, it is towards a global optimal.

5.5 Genes and GA for RESC Algorithm

5.5.1 Point Indices Set

In applying GA to RESC algorithm, we use as a "chromosome" the point set **X** with p points, rather than parameter vector **P**. This is illustrated in Figure 5.7. In the figure, we extract a circle from the points. Three points determine a circle. One chromosome contains three points. Figure Set1 and Set2 show two sample sets of chromosomes ($P_1P_2P_3$ and $P_4P_5P_6$). The circle determined by them is not well fitted to the data. Assume a crossover operator takes the above two chromosomes as parents and generates a new chromosome $P_1P_2P_6$. This new chromosome is better fitted to the data than its two parents.

5.5.2 Gene expression

In our case, each gene is an index to the points of the current processing region. The point index is simply an integer and the whole optimization process is to select the number and the number combinations. We call such gene expression integer gene, or i gene for short. How should we express such integer? The normal approach is to grey code the number and treat each binary digit as a gene (b gene). An example of the binary gene for a three gene chromosome is in Table 5.1 (the example is simple binary code, not grey code).

If we express our genes as binary digits, a chromosome will be a concatenated binary strings. Now we analyze the situation to express each i gene as binary digit.

Figure 5.7 How GA works

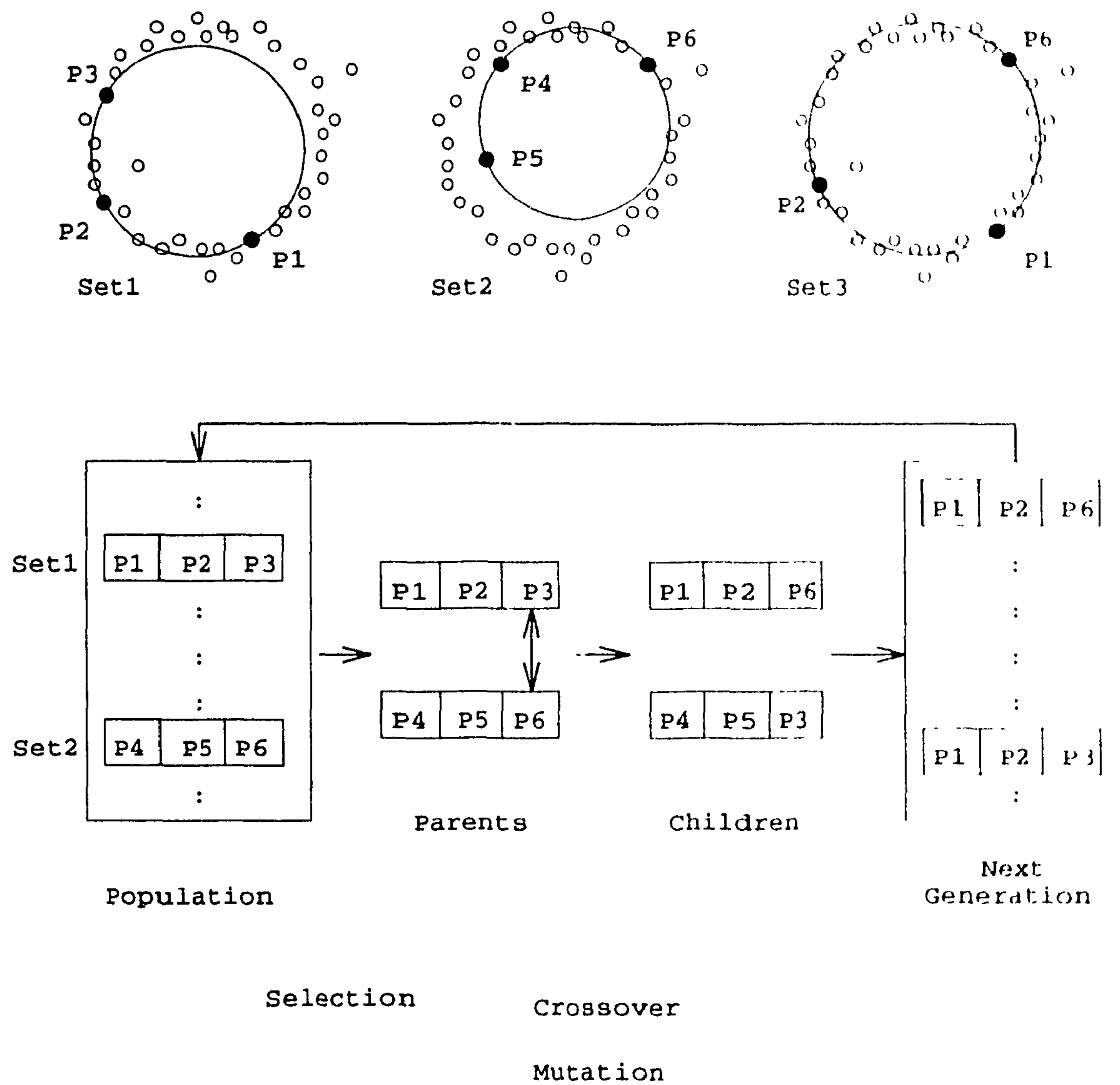


Table 5.1: An example of gene expressions

integer gene	binary gene
571 38 288	1000111011-0000111000-0100100000

The position where two i-genes are concatenated is called the boundary of i-gene. Crossover operation exchanges parts of the parents. If the crossover takes place at the boundary of i-genes, it does not break i-genes, it simply changes combinations of the i-genes in the parents. Otherwise, except for the combination change, it also breaks an i-gene, giving a new number for the i-gene. This is equivalent to a mutation operation for the i-gene.

We will calculate the probability of such break of an i-gene when a crossover operator is applied. Since breaking an i-gene is equivalent to having a mutation on the i-gene, we will also calculate the equivalent mutation rate. Assume the length of a binary string for an i-gene is q and let a chromosome contain c i-genes. Assume that n point crossover operator $(0 < n \leq cq)$ is used and, to simplify the calculation, assume that the crossover may take place at the same position more than once for n point crossover operator $(n > 1)$.

There are $c - 1$ positions where the crossover does not break any i-gene. The probability that n crossovers take place at such positions is: $((c-1)/(cq-1))^n$. Therefore, the probability that a n-point crossover breaks an i-gene is:

$$p = 1 - (\frac{c-1}{cq-1})^n. \tag{5.5}$$

If $q = 10$ (the integer number has domain [0, 1023]) and $c = 3$ (planar surface), for a one point crossover, such probability is 93.1%.

We are more interested in calculating the equivalent mutation rate. For each i-gene, if there is no crossover inside the i-gene's binary string ($q - 1$ positions), then there is no mutation. There are totally $cq - 1 - (q - 1) = q(c - 1)$ such positions. The

Table 5.2: 1-gene breaking probability and equivalent mutation rate

	$c = 3$		$c = 9$	
n	p	m	p	m
1	93%	31%	91%	10%
2	99%	52%	99%	19%
4	100%	67%	100%	27%
15	100%	100%	100%	80%
45	N/A		100%	99%

probability that all n crossovers take place at such positions is $(q(c-1)/(cq-1))^n$. Therefore, the mutation rate (the crossovers does not take place at these positions) is:

$$m = 1 - \left(\frac{q(c-1)}{cq-1}\right)^n. \quad (5.6)$$

Some examples of the mutation rate for $q = 10$ are listed in Table 5.2. Most researchers use 1-point, 2-point or uniform crossover operators. A uniform crossover is equivalent to a n-point crossover where n is equal to half of the chromosome length. In such binary expression, a uniform crossover is equivalent to 15-point crossover for $c = 3$ and 45-point crossover for $c = 9$.

The equivalent mutation rate is very high for such binary expressed genes as listed in Table 5.2. It seems that we have lost control at mutation. If we constrain crossover to take place only at the boundary of genes, such crossover is equivalent to represent the number directly by integer. It is not necessary to represent genes in binary digits, therefore, we do not use binary representation for our genes.

5.5.3 Genetic Algorithm in RESC

Initial population is created by random sampling method until the size of the population U reaches a given level. The point set **X** (chromosome) stored in U is sorted according to its performance, i.e. the histogram power v. The reproduction selects **X** with the probability proportional to its rank in U. The mutation operator replaces each point in **X** with probability M with a randomly chosen point in the current processing region. The process continues until either performance of the population is stable (i.e., the maximum and minimum performance is nearly the same), or the number of offspring generated reaches a limit.

All chromosomes in population U are valid as described previously. After genetic operator is applied to these chromosomes, we have to check the validation again. Invalid offsprings should be discarded.

5.5.4 Differences from the Traveling Salesman Problem (TSP)

The problem solved here is different from the traveling salesman problem (TSP). In TSP, a salesman must make a complete tour of a given set of cities in order that minimizes his total travel distance. TSP is a permutation problem on all cities. In our primitive extraction problem, we extract a few points from all possible candidates. The order of the extracted points is irrelevant since the primitive does not depend on the order. Primitive extraction is to find appropriate points which determine a best fitting primitive to all input points. Therefore, it is not a permutation problem, but a combination problem.

Table 5.3: Synthetic data used in the experiments

Case	Type	Equation	σ	Outlier
1	Ellipsoid	$0.01x^2 + 0.01y^2 + 0.02z^2 = 1$	0.01	10%
2	Two	$0.1x^2 + 0.03y^2 + 0.02z^2 + 0.3x + 0.1y - 0.5 = 1$	0.02	5%
	Ellipsoids	$0.01x^2 + 0.05y^2 + 0.1z^2 - 0.3x - 0.5y + 1.0 = 1$	0.01	5%
3	Plane	$z = 3 + 0.3x + 0.5y$	1	60%

5.6 Experimental Determination of Parameter Settings for Genetic Algorithm

We performed extensive experiments on GA, RESC and real range image segmentation and fitting. Our program implementing the RESC and GA algorithm is written in C and is tested on the Silicon Graphics GTX 220 computer with CPU speed of 20 MIPS.

5.6.1 GA Experimental Design

The parameter setting is very important to make GA work well. Since previous experiments by other researchers [16, 36, 73, 11] on GA control parameter settings were only done for binary coded genes, consisting of 0s and 1s, it was necessary to carry out extensive experiments to explore the performances of GA for RESC method under different conditions.

Several synthetic data cases are used in the experiment as shown in Table 5.3 and Figure 5.8, Figure 5.9 and Figure 5.10. The synthetic data are generated in the domain of $(-10 \leq x, y \leq 10)$ for case 1 and 3 and $(-20 \leq x, y \leq 20)$ for case 2. It is very hard to see the original plane from the picture when it is contaminated by

60% outliers. It is impossible for a non-robust estimation method to obtain correct surface primitives due to a large number of outliers. There are two surface patches in the second case. This means that there are two local optimals and since all data are input to the RESC, it must be highly robust to extract only one patch and consider the other patches as outliers. The experiments demonstrate not only the global optimization of GA, but also the robustness of RESC algorithms. The RESC has correctly estimated all surface parameters in the experiments. More experiments on robustness of RESC method are given in Chapter 7.

As we mentioned before, GA is the stochastic optimization algorithm. To obtain a relative stable solution in order to compare different GA parameter settings, two measures were used in the literature [16, 73]. An *online average* is simply the average of performance of all chromosomes tested during the search. An *offline average* is the average of the best performance for several runs. In each run of the GA, the best performance is the highest histogram power ψ (Equation 4.31) of the current search. Since for our purpose only the best performance is used for primitive extraction, an offline average is our measure for GA. Each set of data is tested 20 times with different seed values for the random function, and final performance is a result of an average over the 20 tests. The control parameter space is as following.

- Population size (S): The population size affects both GA's performance and the overall efficiency. A small population size may not contain enough information for GA to play with. A large size may contain too many weak chromosomes and slow down the convergence. In our experiments, the variations of the population size are 3, 5, 10, 20, 30, 50, 70, 90, 110, 130, 150 and 170.

- Crossover rate (X). This is only used in GA1. The higher the rate, the more new chromosomes are generated in each generation. X is set to: 0.5, 0.7 and 0.9 in GA1 experiments.

- Mutation rate (M). Mutation increases the variability of the population. Higher mutation rate means for each new child a higher chance to incorporate new

Figure 5.8: An ellipsoid with 10% outliers (case 1)

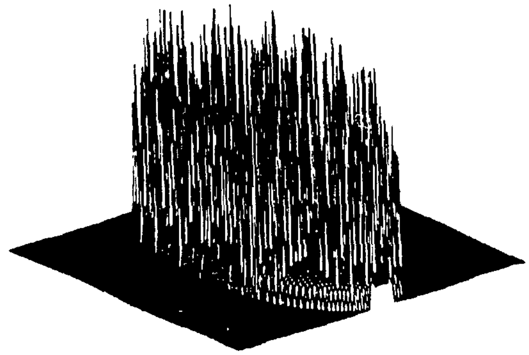


Figure 5.9: Two ellipsoids with 5% outlier (case 2)

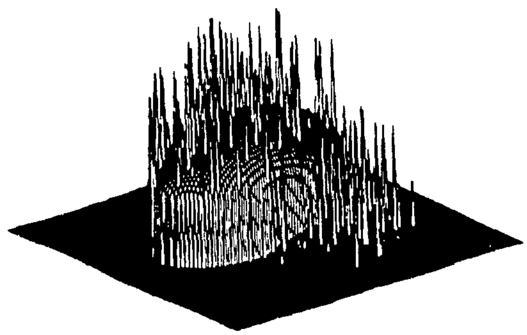
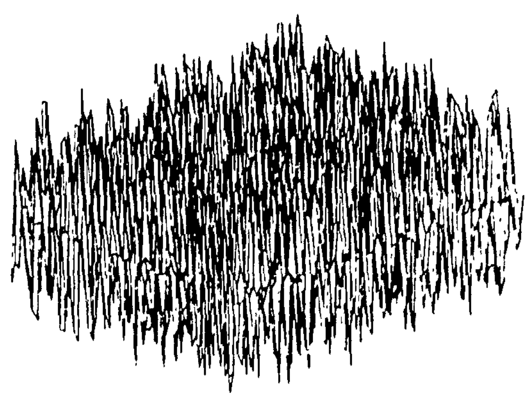


Figure 5.10: A plane with 60% outliers (case 3)



points in the population. In experiments, M is set to: 0.0001, 0.0005, 0.001, 0.0025, 0.005, 0.0075, 0.01, 0.05, 0.1, 0.2 and 0.5. Since it is difficult to display too many lines in one picture, we simply plot part of the results with different mutation rates.

- Number of offspring (N): This is a condition to stop GA. It is obvious that the larger the value of N, the better the approximate optimal solution. A large N could also make GA too slow to be practical. We tested only two cases: $N = 2000$ and $N = 10000$.

We denote a specific GA1 by a triple GA1(S, X, M), and a specific GA2 by two tuples GA2(S, M). A standard GA by De Jong [16] can be expressed as GA1(50, 0.6, 0.001). Suggested parameter ranges in [73] are GA1(20–30, 0.75–0.95, 0.005–0.01).

5.6.2 Experiments and Analysis of GA Parameter Settings

Figure 5.11 through Figure 5.21 are results of experimenting with various parameter settings in different cases. The performance in these figures is defined as the histogram power. As explained before, we simply plot part of the results with different mutation rates, since it is difficult to display too many lines in one picture. All best settings are listed in Table 5.1.

We are not just interested in finding optimal parameter settings from the experiments, we are also interested in finding the rules of the GA performance with various settings. We can draw the following conclusions from the experiments:

- The larger the value of N, the better the performance and the more stable the results. Large N makes the results less sensitive to parameter settings, giving a nearly saturated population, but it could also slow down GA processing.

Figure 5.11: GA1 performance, crossover rate 50% (case 1, 2000 offspring)

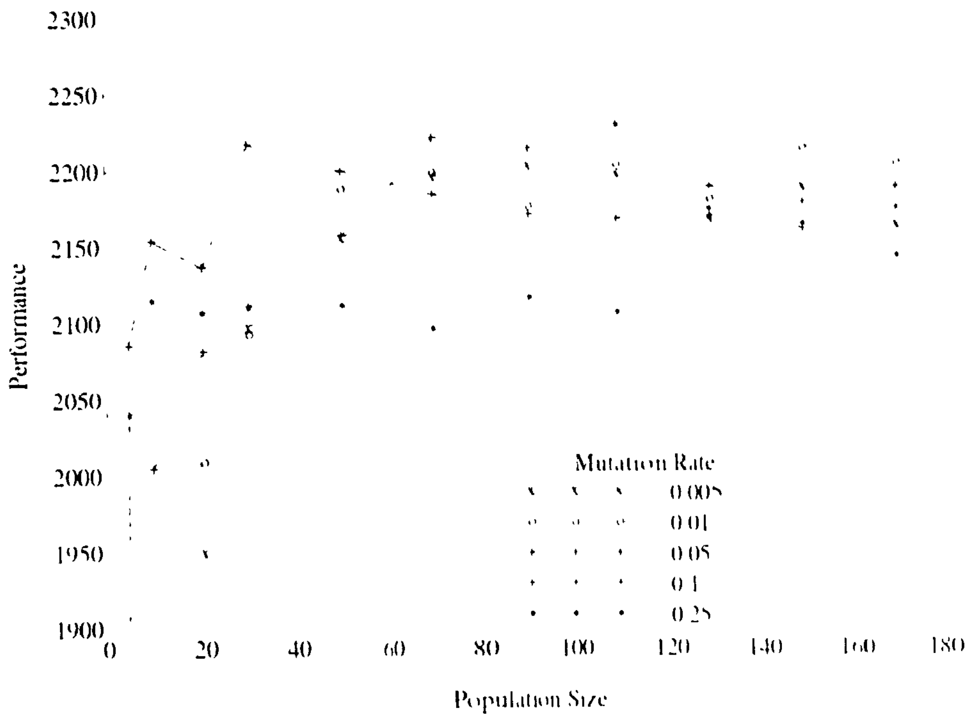


Figure 5.12: GA1 performance, crossover rate 70% (case 1, 2000 offspring)

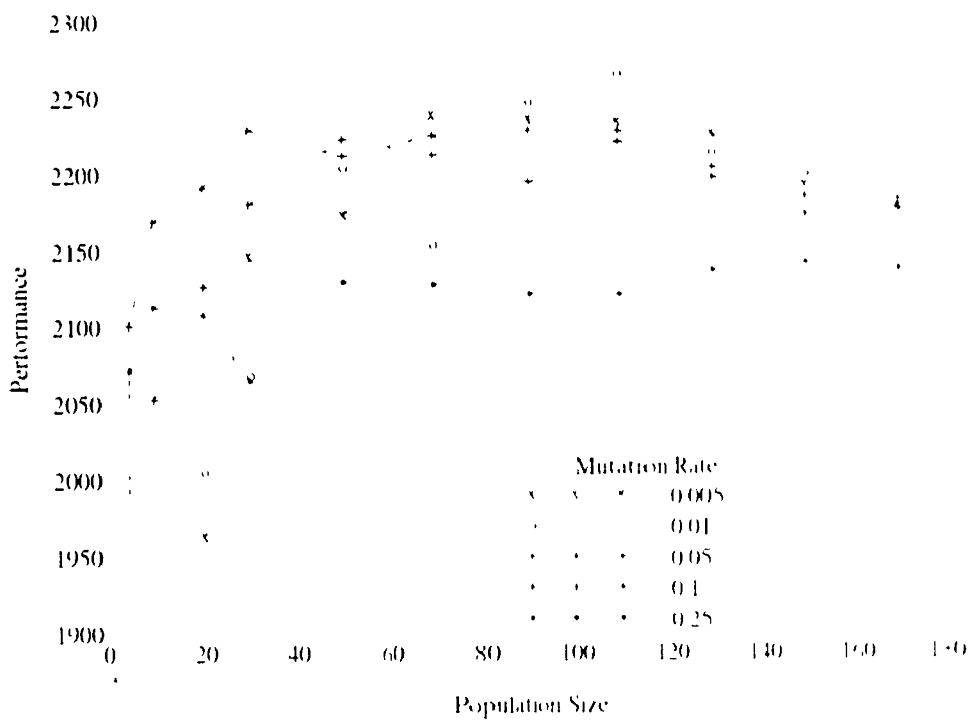


Figure 5.13. GA1 performance, crossover rate 90% (case 1, 2000 offspring)

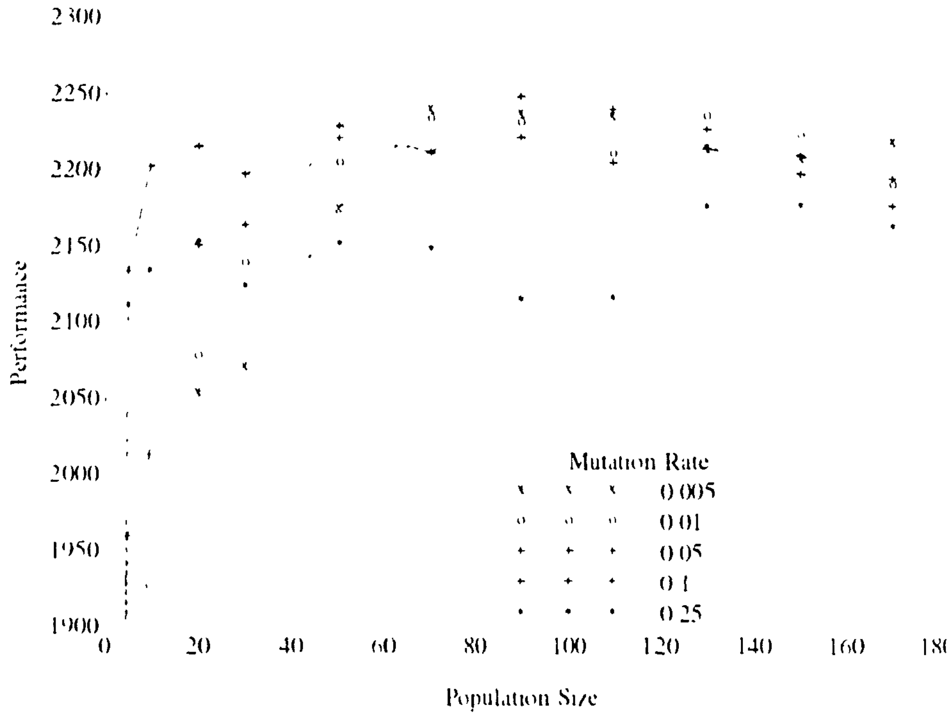


Figure 5.14: GA1 performance, crossover rate 50% (case 1, 10000 offspring)

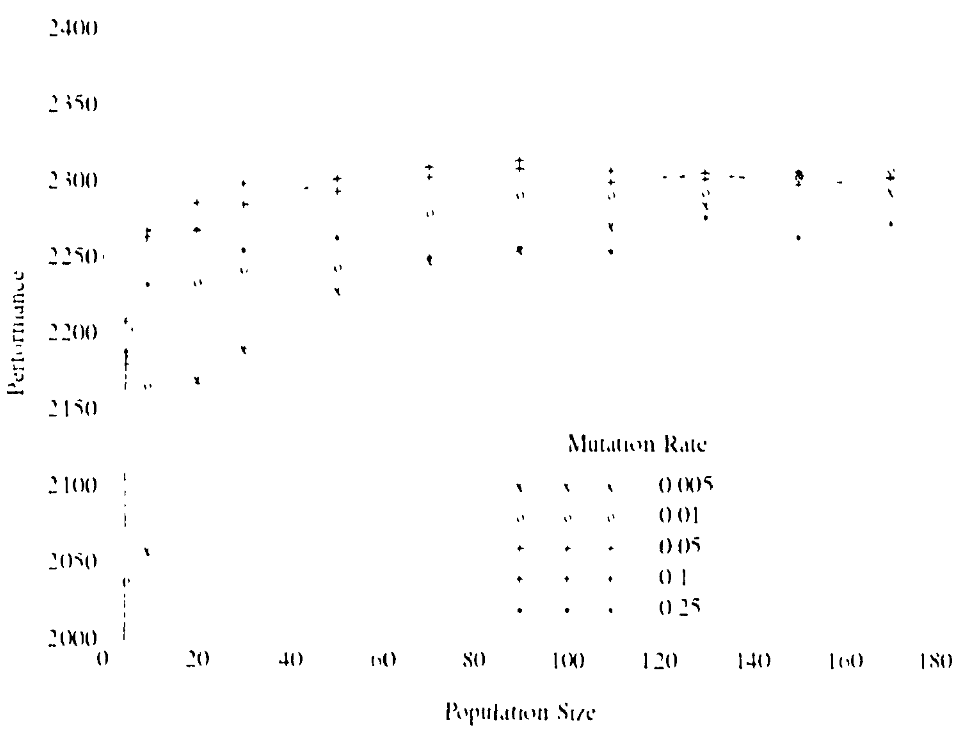


Figure 5.15: GA1 performance, crossover rate 70% (case 1, 10000 offspring)

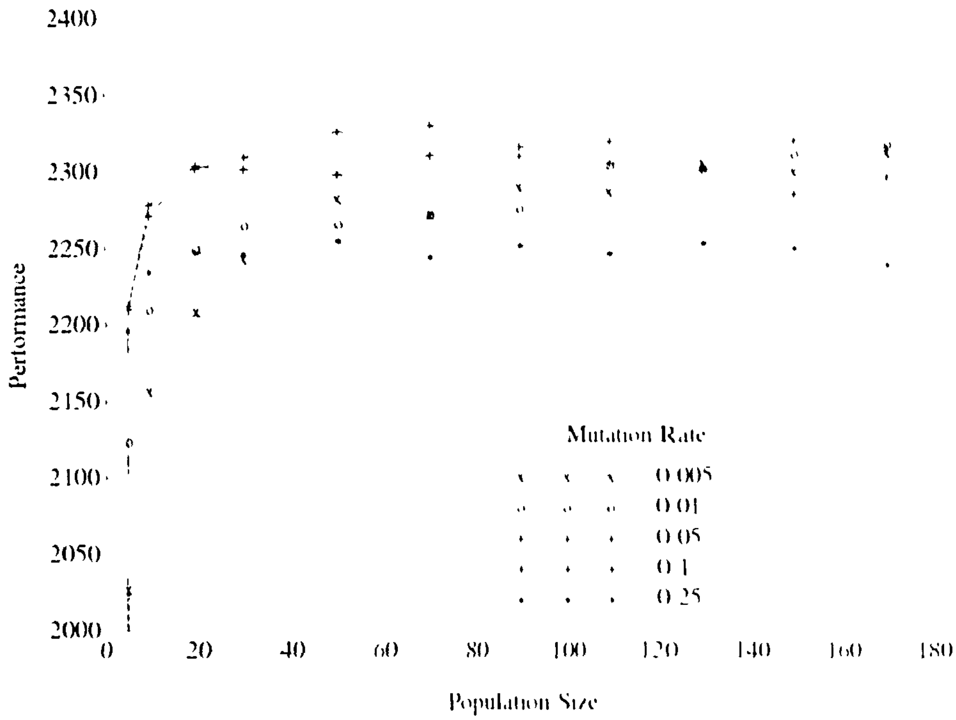


Figure 5.16: GA1 performance, crossover rate 90% (case 1, 10000 offspring)

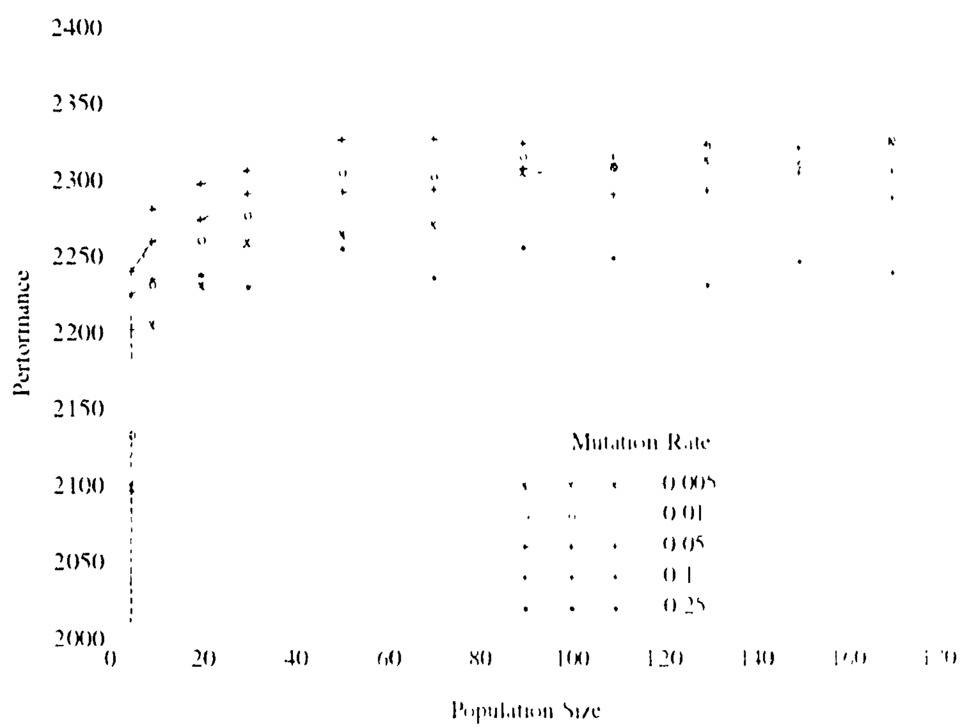


Figure 5.17: GA2 performance (case 1, 2000 offspring)

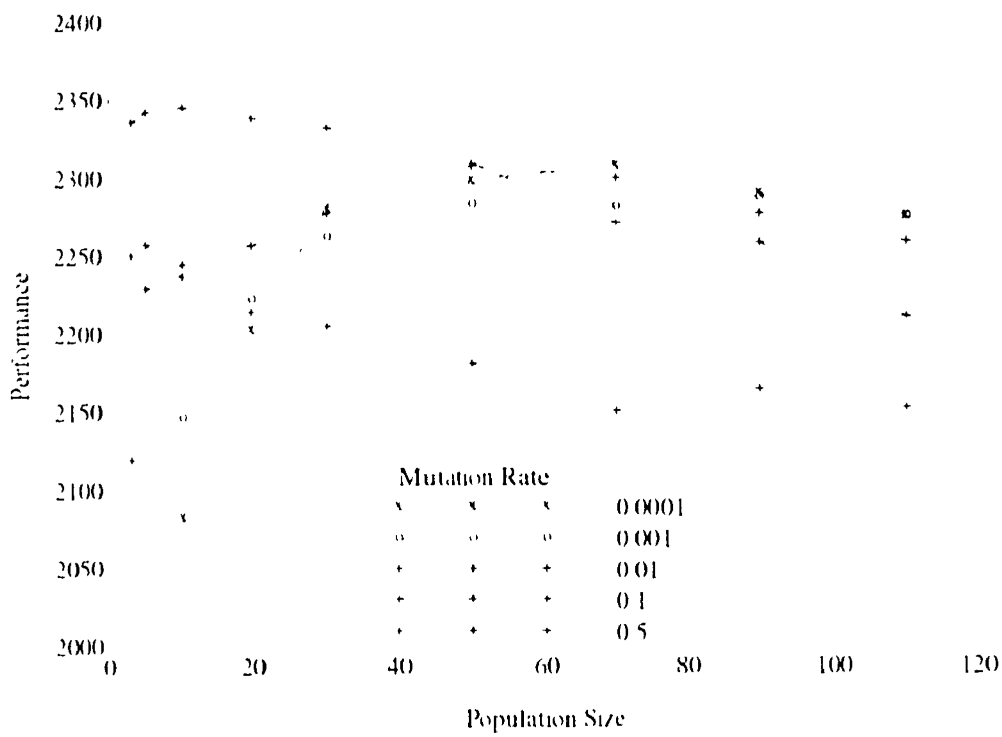


Figure 5.18: GA2 performance (case 1, 10000 offspring)

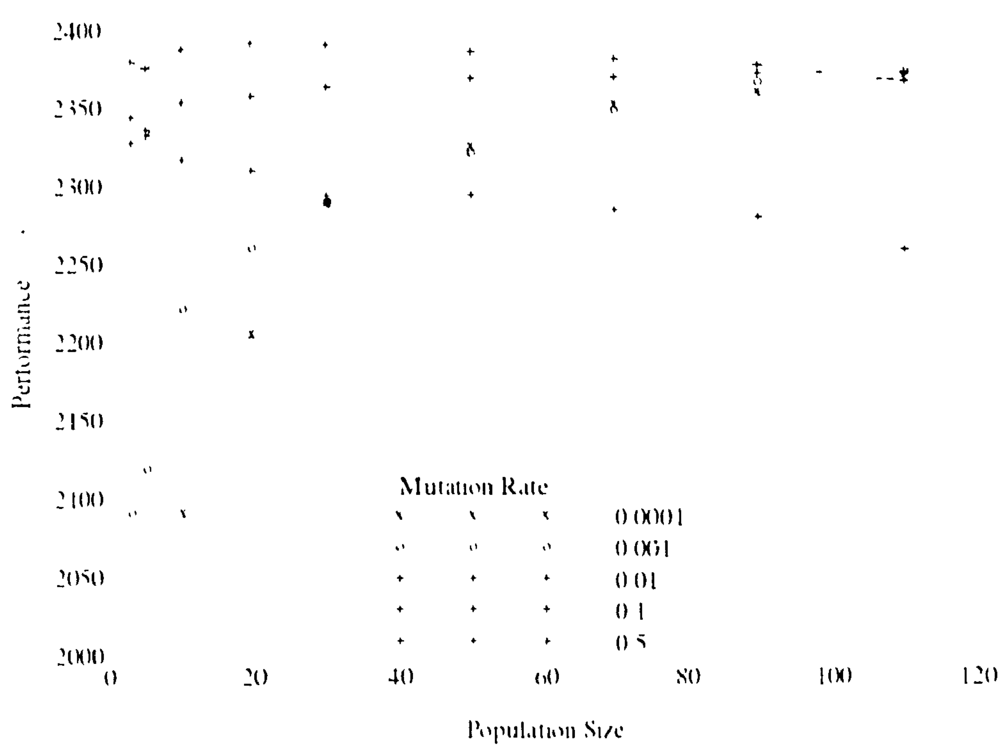


Figure 5.19: GA2 performance (case 2, 2000 offspring)

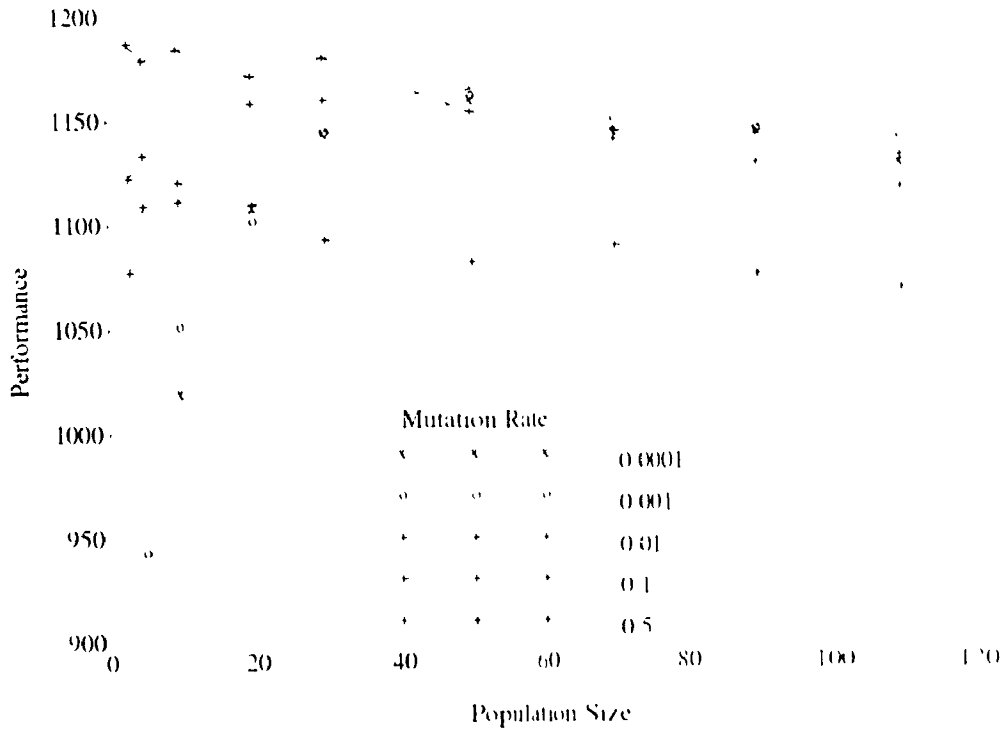


Figure 5.20: GA2 performance (case 2, 10000 offspring)

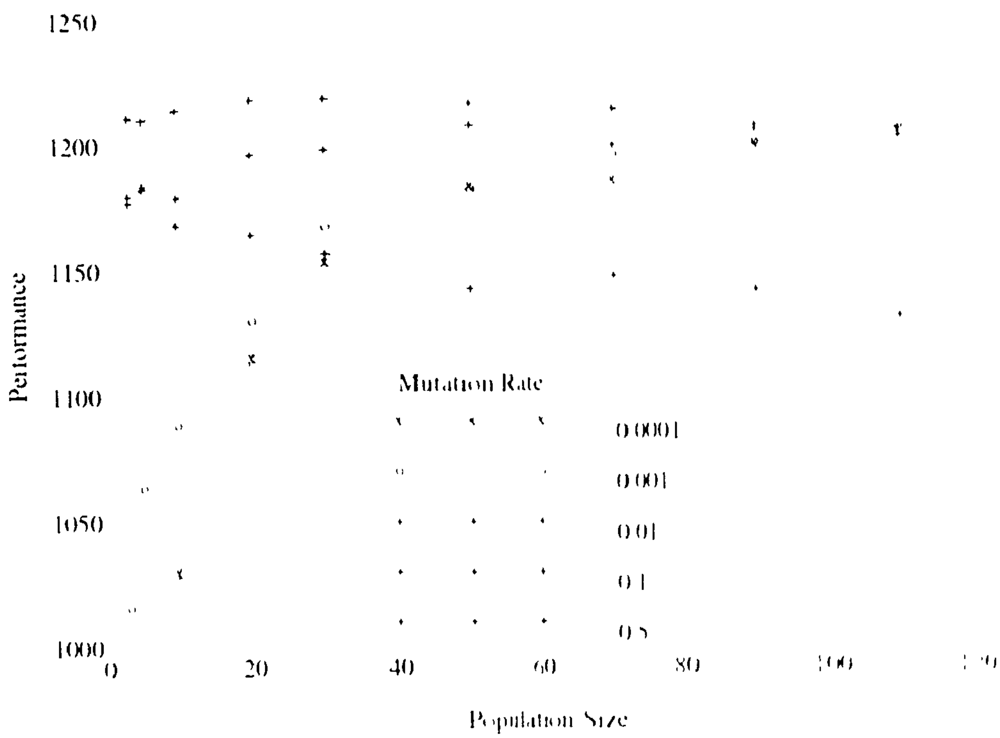
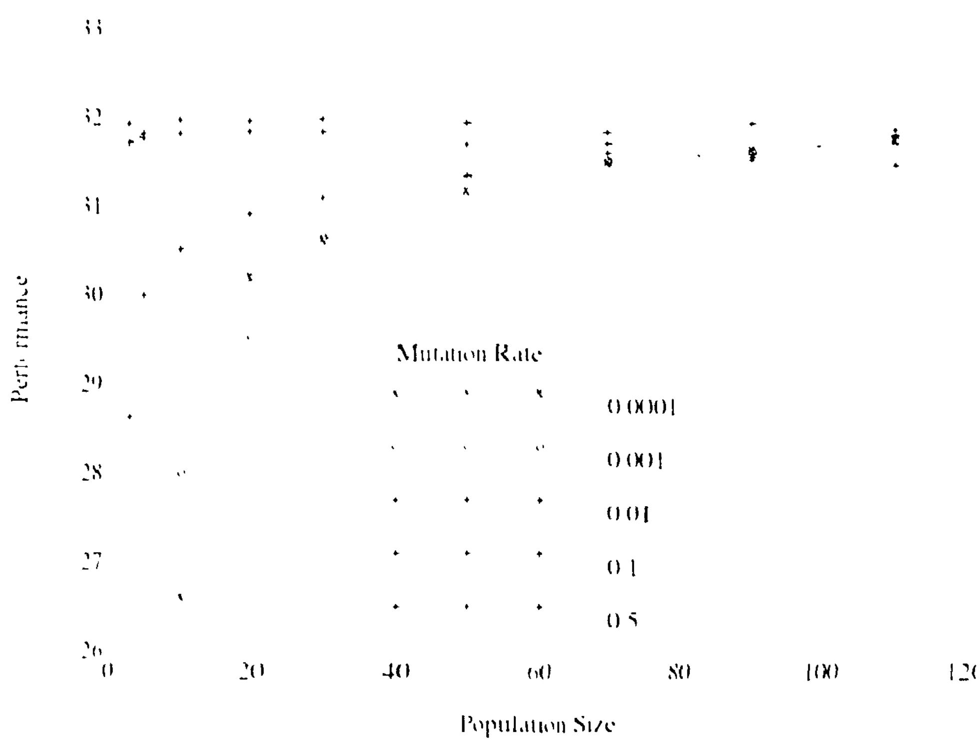


Table 5.4: The best GA settings in different cases

Case	N	c^*	settings
1	2000	2265	GA1(110, 0.7, 0.01)
	10000	2331	GA1(150, 0.9, 0.01)
	2000	2345	GA2(10, 0.1)
	10000	2392	GA2(20, 0.1)
2	2000	1187	GA2(3, 0.1)
	10000	1218	GA2(20, 0.1)
3	2000	32.1	GA2(30, 0.25)

Figure 5.21. GA2 performance (case 3, 2000 offspring)



- By comparing the best settings with the standard one, the population size is larger than that with standard setting and suggested range in all GA1 tests.

- Comparing GA1 and GA2, we find that the steady state GA2 approach is better than the generational replacement GA1. The likely reasons may be as follows. In GA2:

  1. the population is updated immediately after a new chromosome is generated and evaluated.
  2. the reproduction selects the newly updated population, giving faster feedback than the generation method.
  3. best performing chromosomes are always retained in U and the worst ones are discarded immediately.

- In GA2, the best settings are with the mutation rate much larger than that in literature, and the population size is much smaller than GA1. In GA2, since all best chromosomes are always kept in the population the higher mutation rate will bring in more new alleles into the population. Small population size makes the average performance of the total population high. Therefore the selected chromosomes from U have relatively high fitness.

- Different parameter settings may result in quite different GA performance. A better setting may yield much better performance for 2000 offspring than poor settings for 10000 offspring.

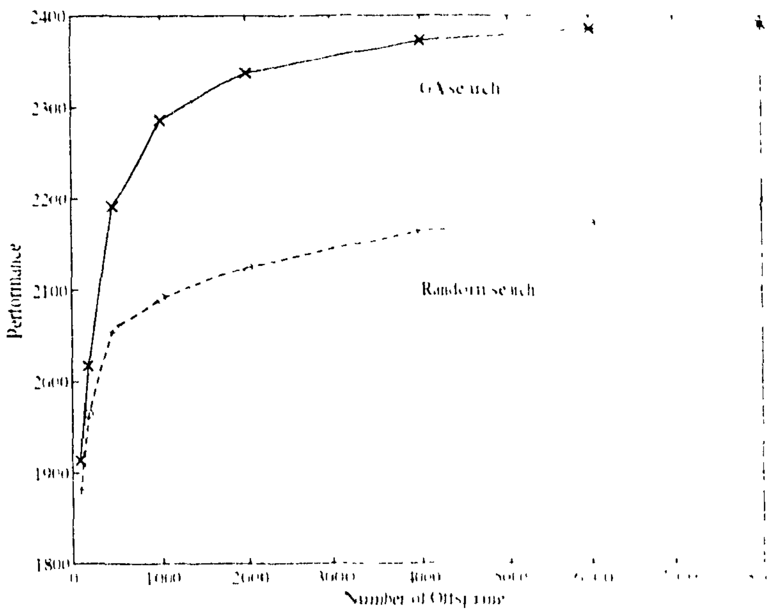
The best settings of our experiments are quite different from those in the literature. In our experiments, the optimal mutation rate turned out to be much higher than the value suggested in the literature. The reason is the representation of the gene. Our gene consists of 1024 alleles at most. Since one chromosome has p genes, the total number of allels T in the population is:

$$T = Sp \tag{5.1}$$

where S is the population size. This total number T contains repeated alleles. The total number of different alleles is less than T. Suppose S ranges from 5 to 100. T ranges from 15 to 900 for $p = 9$ and ranges from 15 to 300 for $p = 3$. In any case, the total number of alleles in the population is less than the total number of alleles in the input data. It means that the total population can not contain all different alleles, it by no means has all possible combinations from these alleles. It is obvious that in optimization process, we must explore other alleles which are not in the initial population. This is why we need higher mutation rate than for a normal binary representation.

Figure 5.22 shows the acceleration of GA over pure random search. As in other experiments, the result is an average of 20 experiments. GA parameters are set to: population size 20 and mutation rate 0.1. The merits of GA are obvious. After generating and evaluating 2000 offspring, GA's search grows slowly, a similar growth rate as that of random search. Therefore, we stop our GA to after 2000 offspring have been generated. We also compared the results using GA for real range image segmentation and fitting. The range image used for the experiment is *The Grip* (Figure 7.27). We limit the maximum number of evaluations for each primitive to 2000. There are some other criterions for stopping the iteration of GA and pure random search, such as the standard deviation of the current patch, the maximum and minimum differences of the performance in the population, *etc.* Even in the pure random search algorithm, we remain the population and the associated operations, such as insert, delete, *etc.*, to keep the amount of computation as similar as possible. The processing time is quite different. The pure random sampling method takes 343 seconds of CPU time whereas the GA takes only 115 seconds. Except the time difference, the segmentation and fitting results are also different. Pure random sampling method obtains less accurate fitting for each primitive although it takes about 3 times longer than the GA does, giving bad segmentation.

Figure 5.22: GA acceleration vs. random search



5.7 Summary

In this chapter, we have briefly introduced genetic algorithms (GA) and explained how to incorporate GA into our RESC algorithm.

Binary and integer expression for a gene is analyzed. Crossover operation breaks, at a high probability, a binary gene expression for an integer, equivalent to having a mutation on the integer. This results a very high mutation rate and we can not control it. Therefore, we do not use a binary expression for a gene. Instead, integer gene, the indices of input points, is used in our GA.

Although there is still no fundamental theory about the performance and convergence of GA, the empirical studies give a guideline for selecting GA parameters. Two different GAs are tested. A steady state GA has much better performance than a generational replacement GA. The experimental results show that a mutation rate is much higher than the range suggested by other researchers. Using integer gene in our GA is one reason for such high mutation rate. Upon a good selection of mutation rate, population size is not a very sensitive factor. GA can work well over a large range of population size. The RESC algorithm works very well under the support of GA for the stochastic search of the best sample points over the unsegmented range images.

Chapter 6

Segmentation

Segmentation is a very important fundamental processing. Chapter 2 extensively reviewed various methods for range image segmentation. We explain our algorithm in detail in this chapter.

The main idea of this chapter were published in the *Proceedings of IEEE 199[illegible] Computer Vision and Pattern Recognition* [86], *Proceedings of the SPIE Advances in Intelligent Robotic Systems and Sensor Fusion IV: Control Paradigms and Data Structures* [83]. It was also submitted to *IEEE Transactions on Pattern Analysis and Machine Intelligence* [87]. We explain the algorithm much more detail than the previous publications.

Section 6.1 outlines the segmentation process. Section 6.2 describes a simple step-edge detector. A preliminary segmentation is performed by it. Section 6.3 [illegible] the segmentation algorithm in detail. Section 6.4 solves the small region problem which is due to outliers or edge regions.

6.1 The Outline of Segmentation

We use a two stage segmentation strategy. A preliminary segmentation is applied to the raw image first. Simple step-edges are detected and used to segment the whole image into several regions. A preliminary segmentation can reduce the amount of computation by RESC method as explained in Subsection 1.2.5.

Primitive extractions by RESC method is applied to each region. With RESC method, we can select the best fitting which is always the largest homogeneous primitive in the current processing area because the probability of choosing points from this region is higher than from others. After a primitive is determined, a segmentation algorithm is used to segment the primitive out of the region. The further processing of the region is then performed in a smaller scale. Our segmentation method differs from those of other authors because in the surface fitting process, we already have the segment implied by fitting the equation which fits a homogeneous region of the surface. Our segmentation algorithm extracts the largest continuous region which this equation fits, in the sense that the residual of each pixel is within a threshold determined during the fitting process.

6.2 Preliminary Segmentation

In subsection 1.2.5 of chapter 1, we mentioned that a preliminary segmentation is necessary to divide a large region into (possibly) several smaller regions in order to accelerate the processing speed. Neighboring pixels with discontinuities are obviously indication of possible edges separating two regions. We use a simple step-edge detector to find edges, i.e., if a pixel satisfies the condition:

$$|x_i - x_{i-1}| > T_x \qquad \text{or} \qquad |y_i - y_{i-1}| > T_y, \tag{6.1}$$

then pixel i is classified as a step edge pixel.

The initial region is formed by marking all four neighbor connected pixels which are not step-edge pixels. Each initial region is marked by a temporary label. The *step-edge region* is a region with slope exceeding the step edge threshold. *Gap regions*, where the range value is not available, as well as step edge regions, are given special labels in order to identify them during segmentation.

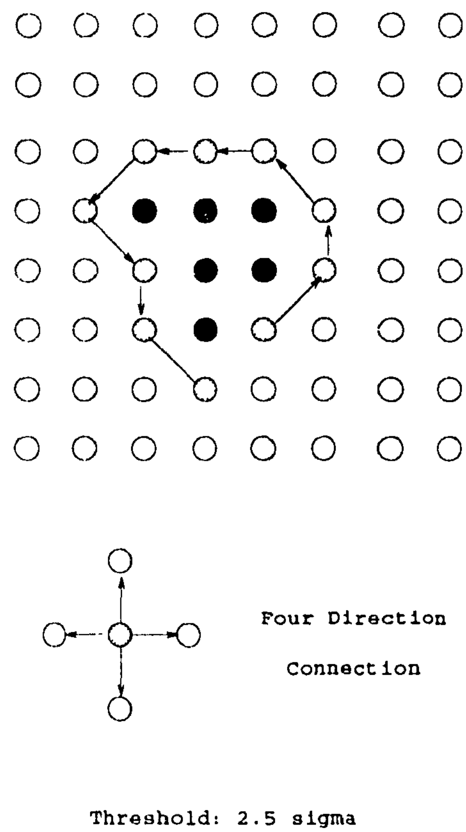
The preliminary segmentation is for reducing the computation load of the RESC algorithm. It is not a necessay step. Without the preliminary segmentation, the range image can still be segmented normally. In fact, one of the examples (see Figure 6.29) in the experiments does not have step edges. The RESC can still extract primitives and segment the range image properly.

6.3 Segmentation Algorithm

Unsegmented regions are labeled by temporary labels in the initialization process, including the preliminary segmentation process performed by a simple step edge detector. Segmentation process will label surface patches with permanent labels. We use a boundary-list method to find a continuous region which is four neighbor connected, allowing a one-pixel outlier. Here is the basic idea of the algorithm:

1. Initially all pixels are unlabeled, and boundary lists L and F are empty.
2. Find a seed whose residual is within the threshold by scanning the unlabeled pixels in the current processing region. Label the seed pixel and put it into boundary list L.
3. Clear list F.
4. Take every element in the boundary list L and check its four neighbors. If a pixel is unlabeled and its residual is within the threshold, label it and put it into boundary list F.

Figure 6.1: Segmentation



5. Switch I and E.

6. Repeat steps 3 and 5 until no elements remain in the lists.

7. Repeat step 2 to step 6 until no unlabeled pixels remain in the current processing region.

8. From the labeled regions, select the one having the largest number of pixels as the result.

We call it boundary list method because the algorithm is based on two lists which keep the current boundaries of the region. The basic idea is shown in Figure 6.1. The details of the algorithm are explained below. The algorithm uses a boundary list method which requires two lists I and E. Each entry in list I contains 4 elements as in Table 6.1.

Table 6.1: Elements in list l

l^p	current position (index of M)
l^d	direction for next step
l^r	residual at current point
l^k	outlier counter

Symbol l_i is the ith entry of the list L. The same structure and the similar expressions (replace l with f) hold for list F. The two lists are switched alternatively during the segmentation process. Four direction connection is used in the algorithm. The whole algorithm is described in Figure 6.2 and Figure 6.3. Figure 6.2 is the segmentation algorithm for finding the largest continuous (one outlier is allowed) region. Figure 6.3 is a search method used repeatedly in Figure 6.2.

The two thresholds in the algorithm can be set according to the standard deviation σ determined by RESC method during the fitting process. In our experiment the thresholds are determined empirically as: $\tau_s = 2.5\sigma$ and $\tau_k = 3$. After each segmentation, an analytical relaxation method [51] is used to compare current segmentation with its neighbor on the boundary and to adjust segmentation if necessary.

In the segmentation process, gap and edge regions are normally partially eroded by fitted surfaces. The edge regions cannot be fitted well because there are only a few points on the surfaces of large slope; also, the range data are not reliable in such situations. If position and orientation between the object and the range sensor are changed, these regions may become larger and have smaller slope and the data would be much more reliable. The edge and gap regions should therefore not be processed

1. Set τ_s as segmentation threshold and τ_e as step edge threshold, and $\tau_s < \tau_e$.
2. Calculate residuals of all points in the processing region according to the parameters determined by RESC algorithm.
3. Find position q in the image where $|r_q| < \sigma$.
4. From point q, we can generate 4 elements in L, $i = 1, \ldots, 4$ with $l_i^r = r_q$, $l_i^c = 0$ and l_i^d in four different directions, respectively. In the following, all 4 new elements are generated accordingly, except for the stated condition.
5. Suppose the original label for current region is η. Define a temporary label ω, which is different from all other labels.
6. Clear list F.
7. Scan each elements i in the list L as described in Figure 6.3.
8. Swap lists F and L.
9. Repeat steps 6 to 8 until no more elements in the list.
10. Repeat steps 3 to 9 to find the largest segment within current processing region.
11. Mark the segment with appropriate label to avoid further processing.

Figure 6.2: The segmentation algorithm

1. Determine a new searching position $p = l^p + l_i^d$. If p is out of the image region, discard l_i.

2. If the label of p is different from η and it is not the edge region label discard l_i.

3. If $|r_p| < \tau_s$, label p to ω and generate 4 new elements in L with $l^s = r_p$

4. If $|r_p| > \tau_e$, point p is a step edge. Do not label p and do not generate any new element.

5. If $\tau_s < |r_p| < \tau_e$ and $l_i^s \neq 0$:

   (a) if r_p has different sign with l_i^s, label p to ω and generate 4 element with $l^s = 0$ (outlier counter).

   (b) otherwise set the 4 new elements with $l^s = 1$

6. If $\tau_s < |r_p| < \tau_e$ and $l_i^s = 1$ then the position p as well as position l^p are considered to be in other segments. Label the positions back to original label η.

Figure 6.3 One round search

1. For each pixel in the small region, check its four neighbor points.
2. If its neighbors are labeled already (it may have 1 to 4 labeled neighbors, and they may have the same or different labels), calculate the residuals with the neighbor equations. Compare the residuals determined by its neighbors and select the one with smallest residual as the point label.
3. If all its neighbors are not labeled, put the pixel in a new small region and process it later.
4. Repeat the steps 1, 2, 3 until no more small region left.

Figure 6.4: Erosion algorithm

6.4 Small Region Handling

The equation for the primitive requires a certain minimum number of data points. A region with fewer pixels cannot be fitted and is therefore treated as a "small region." Normally, small regions occur either at the boundary or inside a region where the residual is greater than the fitting threshold. If regions are to small to be fitted, we consider them as outliers, since the shape with too few pixels does not have enough information for further processing and those regions should be eliminated. We use the algorithm described in Figure 6.4 to erode them. After running the erosion algorithm, surface patches become less fragmented.

6.5 Summary

In this chapter, we have described in detail a preliminary segmentation algorithm, the segmentation algorithm and the erosion algorithm. The preliminary segmen tation can reduce the computation load of RESC method and speed up the whole processing. By the robust RESC algorithm, each region is fitted with first or second order primitive and the noise level (standard deviation) is well estimated. The segmentation algorithm will select a largest region in which all residuals are within the estimated noise level. The segmentation algorithm uses two lists which store boundaries of the processing region. Four-neighbor connectivity is used. The segmentation process is easy when all required information is present. The segmentation algorithm in this chapter is well developed, thoroughly tested, and works efficiently.

Chapter 7

Experiments of RESC method and Range Image Segmentation

We have empirically tested the proposed RESC method. The experiments were first performed on two dimensional synthetic data and profiles of range image because it is easier to demonstrate the results of fitting and segmentation, therefore it is easier to develop and test the algorithm. After two-dimensional experiments, we expand the algorithm to three dimensional cases. The RESC algorithm part is the same, but the random sampling method of a two-dimensional region and the segmentation algorithm are different.

The main results of this chapter were published in the *Proceedings of IEEE 1992 Computer Vision and Pattern Recognition* [86], *Proceedings of the SPIE Advances in Intelligent Robotic Systems, Sensor Fusion IV: Control Paradigms and Data Structures* [83] and *Proceedings of the Canadian Conference on Electrical and Computer Engineering* [84]. It was also submitted to *IEEE Transactions on Pattern Analysis* *and Machine Intelligence* [87].

Section 7.1 briefly introduces the experimental environment and lists the user modifiable parameters of the experimental algorithm. All two dimensional experiments are explained in section 7.2 and three dimensional in section 7.3. In synthetic data experiments we control the noise levels, the number of pixels, the shape of images, *etc.* It may clearly demonstrate the performance of the algorithm in various conditions. Real data experiments can verify the conclusions from the synthetic ones and demonstrate the usefulness of the algorithm in real situations. We have performed both synthetic data experiments and real range image data experiments in sections 7.2 and 7.3. Section 7.4 summaries the chapter.

7.1 Experimental Environment

Our program implementing the RESC algorithm is written in C and C++ and implemented on the Silicon Graphics GTX 220 computer with CPU speed of 20 MIPS. Part of the experiments have time counting. Although the computer has two CPUs, our algorithm is a serial one without using the parallel processing capability. In the future, we may parallelize our algorithm in order to speed it up further. The Gaussian noises are generated with the IMSL library on VAX6510. Most illustration figures are generated with the MATLAB software package.

All user-modifiable parameters of the segmentation and fitting algorithm are listed in Table 7.1. Each symbol and its meaning can be found in related chapter.

We do not have a high quality laser range finder in our laboratory. We use the range images provided by the Photonics and Sensors Section in Division of Electrical Engineering at the National Research Council of Canada (NRC) and the Pattern Recognition and Image Processing Laboratory of Michigan State University (MSU PRIP Lab) in public network domain. All range images are rendered by technique

Table 7.1: Parameter values in experiments

Name	Symbol	Value
Determinant validation	ϵ	10^{-10}
Compressed Histogram	δ_{finest}	0.05
	H_n	2000
	ρ	0.12
	α	1.3
	β	1.0
	ξ	0.88
Primitive order switch	T_p	10^{-3}
Number of pixels in a region	R_n	1000
Genetic algorithm	S	10
	N	2000
	M	0.1
Segmentation algorithm	τ_s	2.5σ
	τ_l	$3\tau_s$

of computer graphics. Although range images have only depth value of each point, we produce quite realistic images with the support of the graphics library of Silicon Graphics computers. The materials, light sources, and lighting models can be defined. We define surfaces of objects in range image as shiny metal surfaces. Some highlights can be seen from the rendered images. The objects in the scene can be scaled, moved or turned on screen so that we can see details of any position of object and can see the object from any view point.

Our implementation of the RESC and genetic algorithm is highly visual able and interactive. Each region of the process range image are colored differently. During the processing of the range image, each random sample point and current best sample point set can be seen dynamically. At any time of the processing, we can pause the process to check values of original image and the fitting surface at the cursor point by click a mouse button. It also shows the Gaussian and mean curvatures of each point and the invariants and pose matrix and quadratic surfaces (see Appendix C). The software system provides a good testbed for range image processing.

In synthetic data experiments (subsection 7.2.1 and 7.3.1), we have investigated the fitting errors of different methods with various noise levels and outlier percentages. Comparisons of the fitting error of each set of parameters of the primitive have been plotted for different methods. In our tests we have used three different method — the least squares (LS) method, the least median squares (LMS) method and the RESC method. For each experiment, we have analyzed the experimental results. In three dimensional synthetic data experiments in section 7.3, we tested average performance of the LMS and RESC methods for planar and quadratic surfaces under various noise levels and outlier percentages. As far as real range images experiments are concerned, we demonstrate in sections 7.2.2 and 7.3.3 the fitting and segmentation process of the RESC method for several images, such as a grip, a space shuttle etc.

7.2 Two-Dimensional Cases

Two-dimensional figures and data points are much easier to plot and easier to understand. Compared to three dimensional surfaces, the computational complexity of two dimensional case is much less than three-dimensional case. The program is therefore easier to test on the two-dimensional frames. We did both synthetic data experiments and real range image profile data experiments.

7.2.1 Synthetic Data Experiments

In the synthetic data experiments, we fit a line into a set of two-dimensional data points. The noise level and outliers can be controlled. The original line equation is:

$$y = A + Bx, \tag{7.1}$$

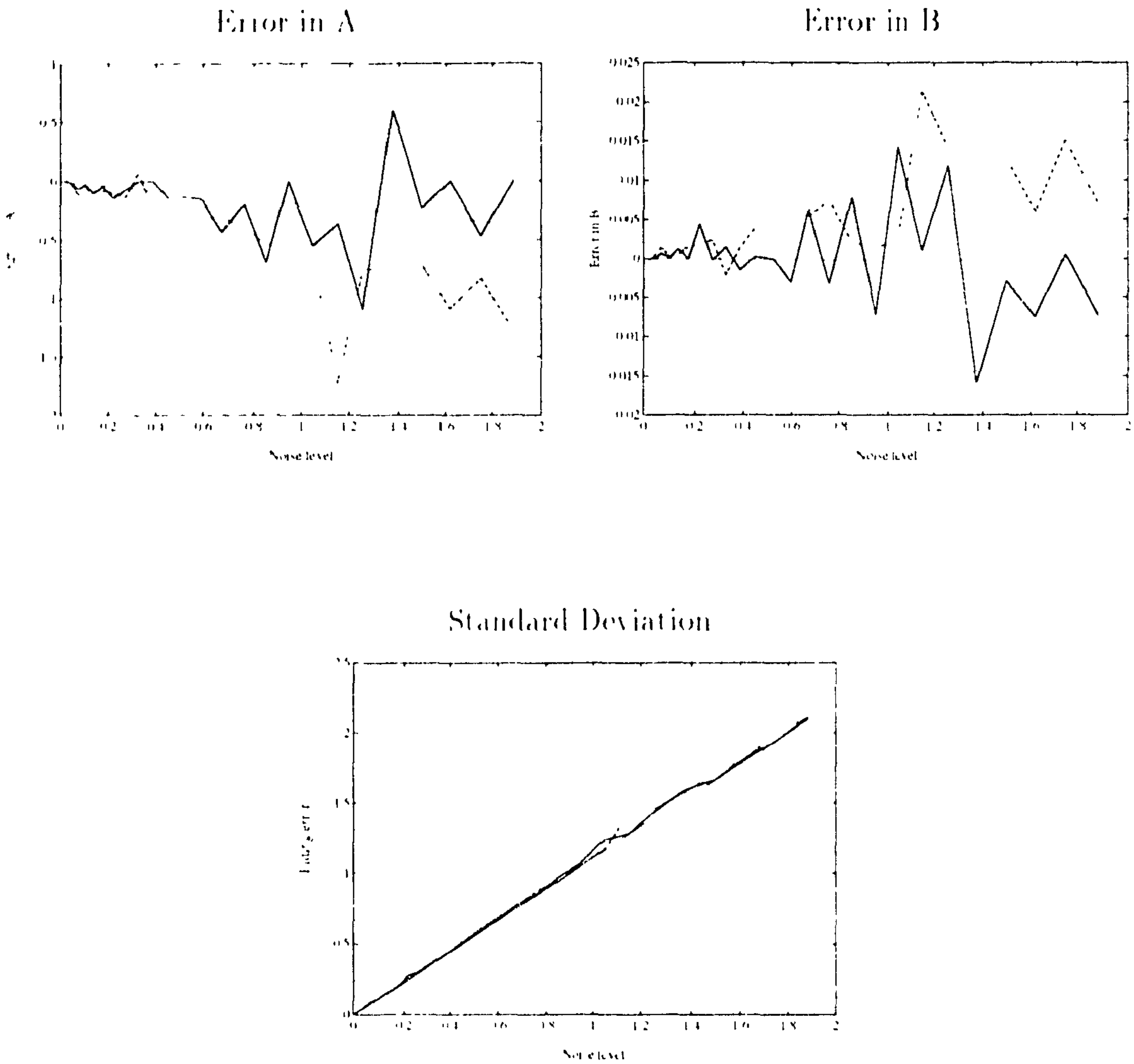
where $A = 2$ and $B = 1.29293$. We compare three-different methods in the fitting process:

1. RESC method,
2. Least median squares (LMS) method, and
3. Least squares (LS) method.

We generate data with 128 points based on equation 7.1. In Figure 7.1 and Figure 7.2, we add Gaussian noise (Equation (4.25)) with standard deviation σ shown in the figure. For each data set, we fit a line using three different method. The differences in parameter values of the fitted equation and the original equation is calculated and shown in different figures. The bottom figures in Figure 7.1 and Figure 7.2 are standard deviation of residuals for data points and fitted equations versus synthetic noise levels. From these figures we can see that the least squares method is the best one for the case of Gaussian noise and without outliers. The least squares method has been proved to be optimal for the Gaussian noise. Therefore, no other method surpass it. RESC and LMS are stochastic methods, therefore, estimated parameter values by the two method fluctuate near the original values. This can be seen clearly in Figure 7.2.

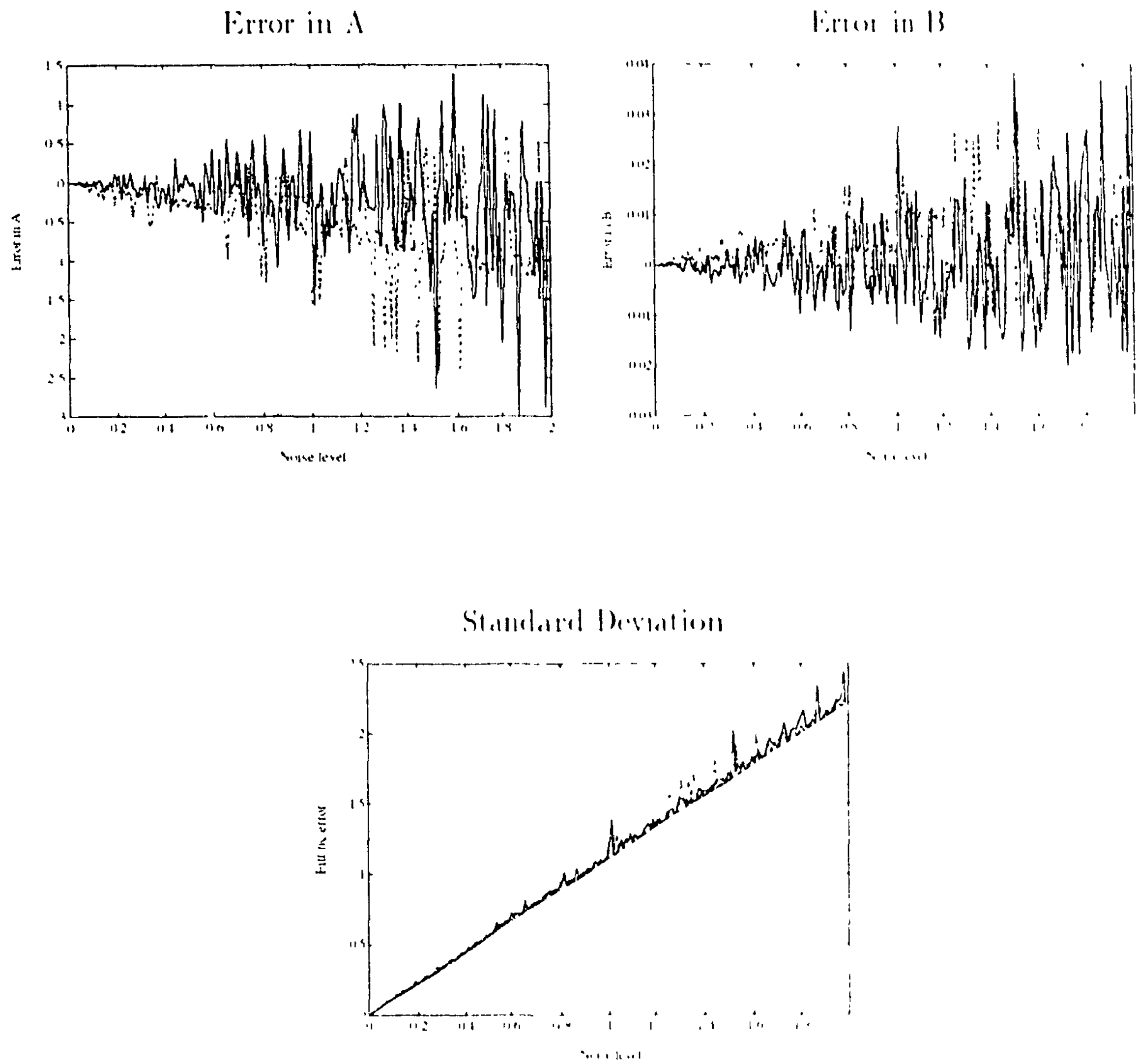
Outliers are added to the data in a given percentage. Outliers are given uniformly in the range of the line and their values are uniformly distributed in the range [−50, 150]. Figure 7.3 shows the case of fitting the synthetic data with outliers and Gaussian noise. In case of outliers, estimation by the least squares method is obviously biased from the original parameter value. The LMS method is good when the number of outliers is less than 50%. RESC method is good even when outliers are near 80% range. Figure 7.4 shows the case of synthetic data with 80% outliers and 0.4 standard deviation Gaussian noise. Only RESC method can tolerate 80% of outliers. In real situations, 80% of outliers like in Figure 7.4 is uncommon. Considering the case in Figure 7.5, we can fit one segment of the data by a line and consider the other segments as outliers. In Figure 7.5, we input the whole set of data points marked by 'o' in the figure to RESC. It succeeds in finding a line segment and treats other segments as outliers. The number of data points belonging to the line segment is about 17% of the total number points in the input data. This demonstrates that RESC can tolerate about 83% outliers. Someone may argue that in Figure 7.5, although the RESC method fits a line to the data, it may also fit a line to other part of the data and it seems no obvious reason to prefer one over the other. It is true that all these six line segments have equal length and the same noise level. Which one is chosen is determined by the highest level consensus (i.e., the histogram power) of the current search. With different seed value for the random number generator, the RESC may fit another segment. In this application, we do not require a unique solution. The criterion is that if we can find a line in this case, we say the method works and is robust. The uniqueness criterion of Rousseeuw and Leroy [69] is not applicable in this case. We elaborate more on the highest breakdown point of a robust estimator

Figure 7.1: Fitting errors vs. Gaussian noise level (line)



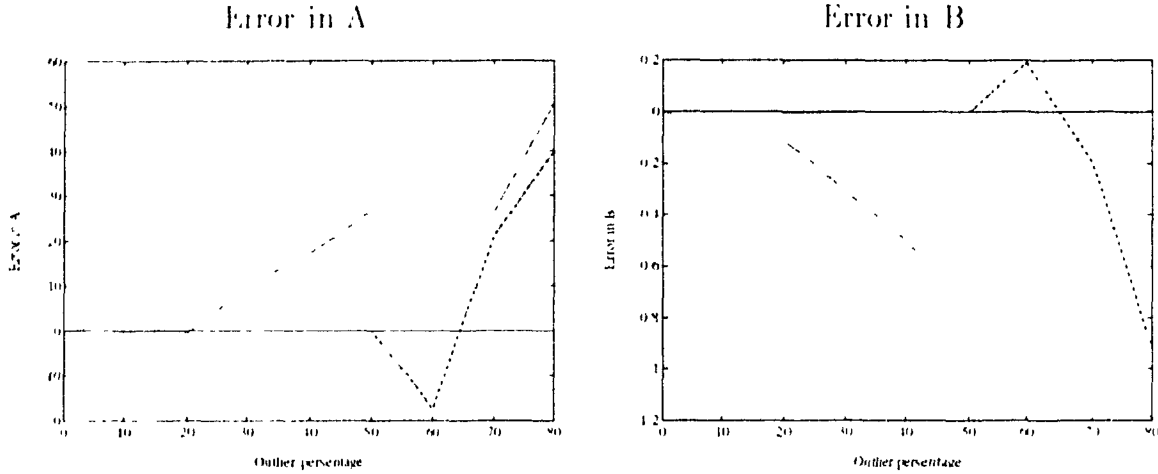
Note: (1) solid line represents errors of the RESC method. (2) dashed line represents errors of the least median squares method: (3) the dotted line represents errors of the least squares method:

Figure 7.2: Fitting errors vs. Gaussian noise level (line, line interval)



Note: (1) solid line represents errors of the RESC method (2) dashed line represents errors of the least median squares method, (3) the dotted line represents errors of the least squares method

Figure 7.3: Fitting errors vs. outliers (line)



Note: (1) solid line represents errors of the RESC method. (2) dashed line represents errors of the least median squares method: (3) the dotted line represents errors of the least squares method:

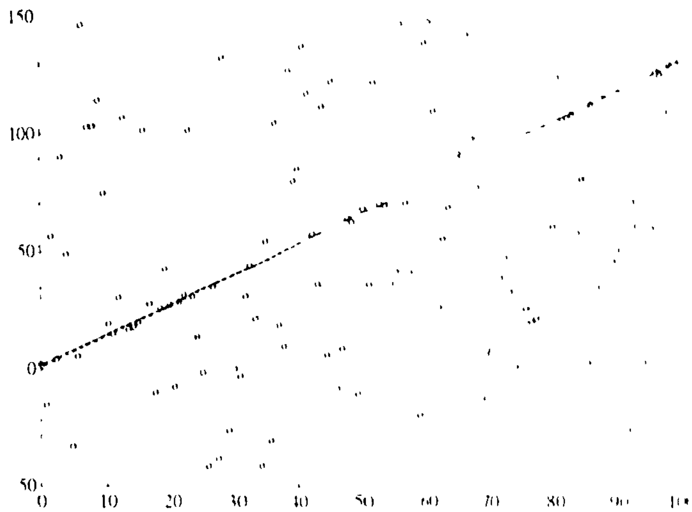
in subsection 7.3.2.

By eliminating points on the line segment and repeating the process on the reduced size data, RESC can segment and fit all the six line segments. Our segmentation and fitting is based upon this consideration. Therefore, a highly robust estimator is essential for such processes. The example of the segmentation of the whole input data of a real range image profile is shown in the next section.

7.2.2 Real Range Profile Data Experiments

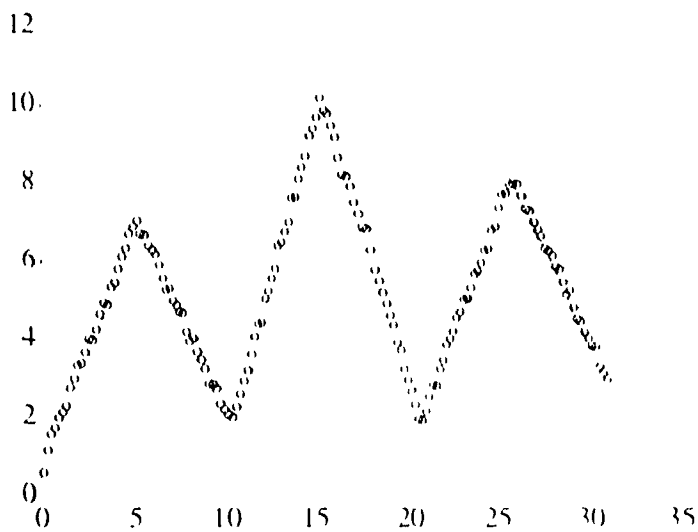
The range image profile data are taken from one line of range image shown in Figure 7.27. The primitives in the 2D profile experiment are straight lines ($F(x,y) = Ax + By + C$, number of sample points $p = 2$) and conic curves ($F(x,y) = Ax^2 + Bxy + Cy^2 + Dx + Ey + G$, $p = 5$). The two-different primitives are switched automatically by the method described before. In the experiments, each segment has its own standard deviation, and they range from 0.02 to 0.12. This demonstrates

Figure 7.4: Fitting a line to the data with 80% of outliers



Note: (1) 'o' represents a data point; data consists of 80% of outliers and 20% of inliers with $\sigma = 0.1$ Gaussian noise. (2) the solid line is the original line. (3) the dashed line is the fitting by RESC method. (4) the dashdot line is the fitting by the least median squares method; (5) the dotted line is the fitting by the least square method;

Figure 7.5: Fitting a line to the data



Note: (1) the 'o' represents a data point; (2) the dotted line is the fitting by the least squares method; (3) the solid line is the fitting by the RESC method

Figure 7.6: Original range image profiles

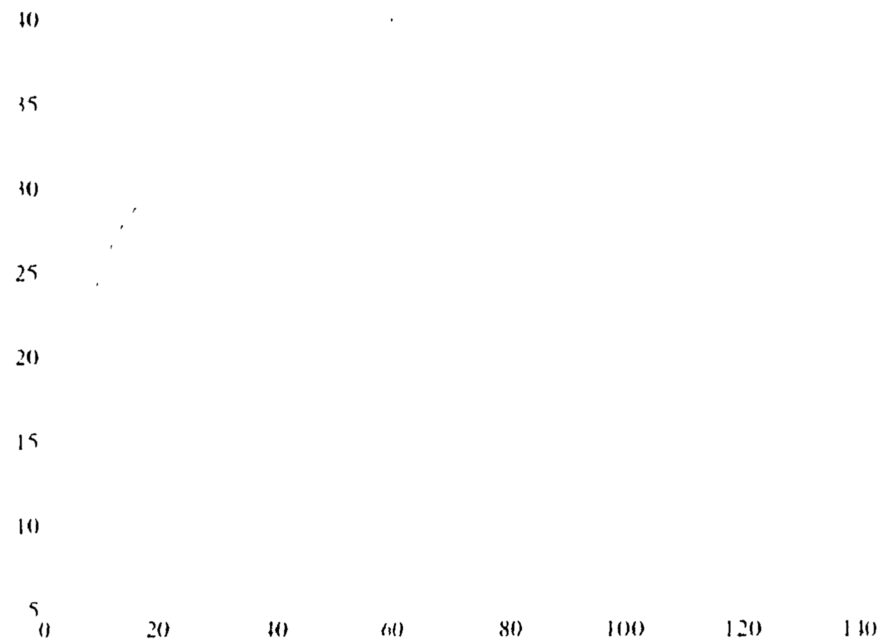


Figure 7.7: Segmentation and fitting with straight lines

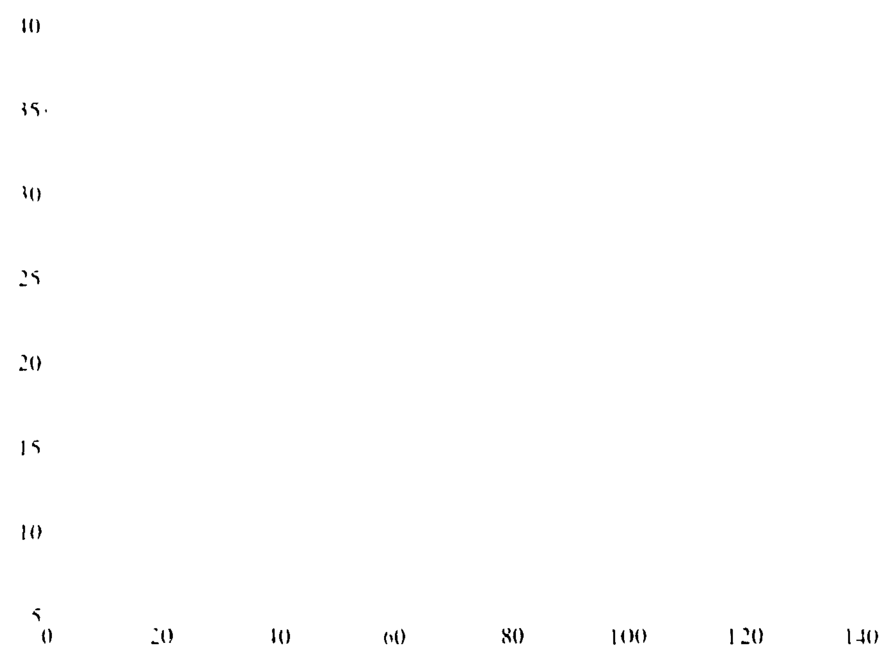
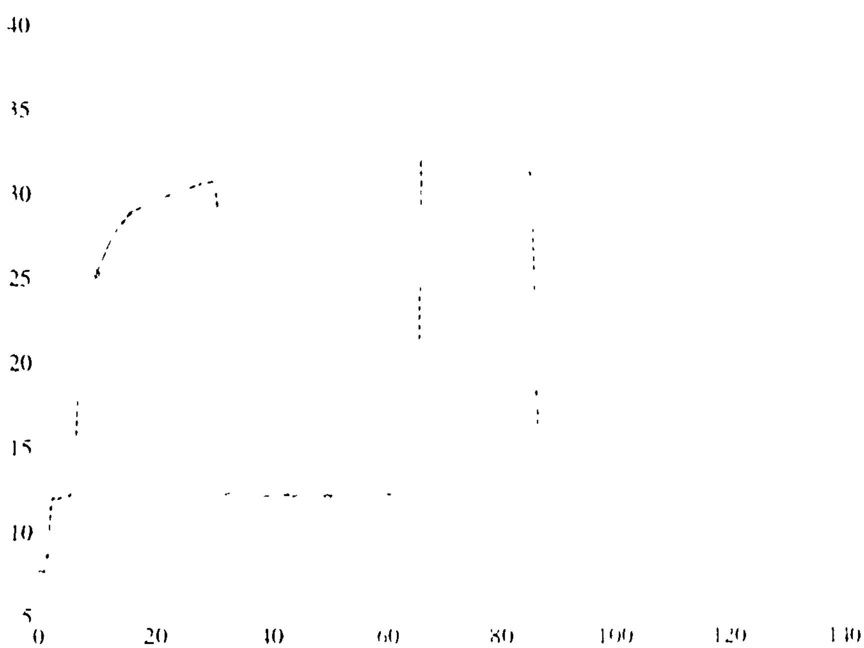


Figure 7.8: Segmentation and fitting with conic curve and straight lines

Figure 7.9: Superimpose of the original and fitted profile



the importance of calculating σ for each segment and the usefulness of compressed histogram (ν ranges from 1 to 12 when $\delta = 0.01$). Figure 7.6 is the original range image profile. We can see the effect of noise which makes the profile rugged. Figure 7.7 is the reconstruction from segmented lines. This is the first stage of segmentation with only the first order primitive. Figure 7.8 is the reconstruction from both the first and second order primitives. We can see that the curved part is much smoother than line fittings. Figure 7.9 is a superimpose of Fig. 7.6 and Fig. 7.8, solid lines represent the original profile and dash lines represent the fitted profile. A symbol 'o' indicates the place where two conic curves join smoothly. Several examples are tested and the results are very encouraging.

7.3 Three-Dimensional Data Experiments

A number of three dimensional experiments have been performed on both synthetic data and real range image data from the Photonics and Sensors Group in Division of Electrical Engineering at the National Research Council of Canada (NRC) and from the Pattern Recognition and Image Processing Laboratory of Michigan State University (MSU PRIP Lab) in public domain (gecko.eecs.wsu.edu, IP address:131.121.32.17 and ftp.ads.com). For synthetic data experiments, various noise levels and outlier percentages have been tested by three-different method: the least squares method, the least median squares method and the RESC method. For real range image data, we demonstrate the fitting and segmentation process of RESC method for several objects, such as grip, space shuttle, *etc.*

7.3.1 Synthetic Data Experiments

In two dimensional synthetic data experiments, we have seen that the results by RESC and LMS slightly deviate from the actual value. This fluctuating nature of the results is due to stochastic process governing both RANSAC and LMS methods. Since three-dimensional experiments are more important to us, we evaluate the experiments by averaging the experimental results to compare the performance of different methods in a statistical sense.

Averaging the Results

To better understand the algorithm's performance, the results should be averaged and evaluated in a statistical sense. We used two ways of averaging the results.

- *Data-average*: generate 30 different sets of synthetic data and run the program once for each set of data. The average is on the 30 results from 30 different sets of data.
- *Run-average*: generate only one set of data and run the program 30 times with different seeds in random number generator. The average is taken over 30 runs for one set of data.

Since data-average tests a number of different sets of data, it is not easy to obtain a stable average. Run-average yields more stable results than data-average. Our comparisons for different methods are mostly based on the run-average.

Planar Primitive

In experiments involving 3D synthetic data, we tested two different primitives. A plane, the first order primitive, has the equation,

$$A + Bx + Cy$$

We generate z values from this equation on a 128×128 grid where $-30 \leq x \leq 30$ and $-10 \leq y \leq 10$. Given x and y values, a z value can be generated from Equation

Figure 7.10: A plane with Gaussian noise ($\sigma = 1$)

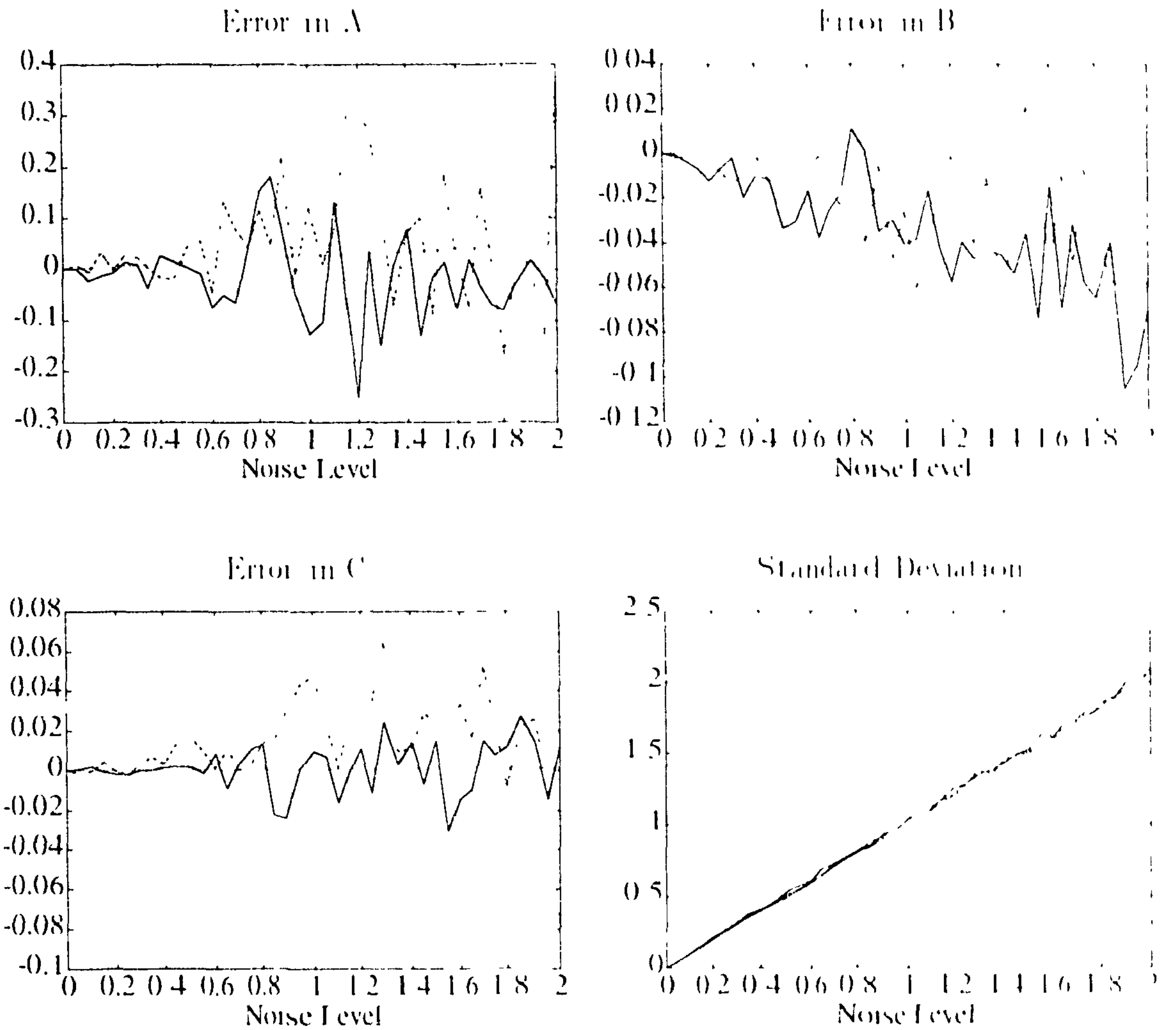
7.2. We use parameters,

$$A = 3, \qquad B = 0.3, \qquad C = 0.5. \tag{7.3}$$

Gaussian noise is added to the z value of the synthetic data. Figure 7.10 shows what the plane looks like when corrupted by Gaussian noise with $\sigma = 1$. Figure 7.11 shows the fitting errors versus the Gaussian noise level. The noise level is σ in equation 4.25. The mean value μ is set to 0. The least squares method is evidently the best for Gaussian noise without outliers. The right bottom figure in Figure 7.11 shows the standard deviation of the residuals calculated from the fitting equation. Figure 7.11 contains only one set of data at different noise levels. Figure 7.12 shows the data average results. The results of run average on 20 runs of the RESC and LMS are shown in Figure 7.13. It seems that no methods superior to others.

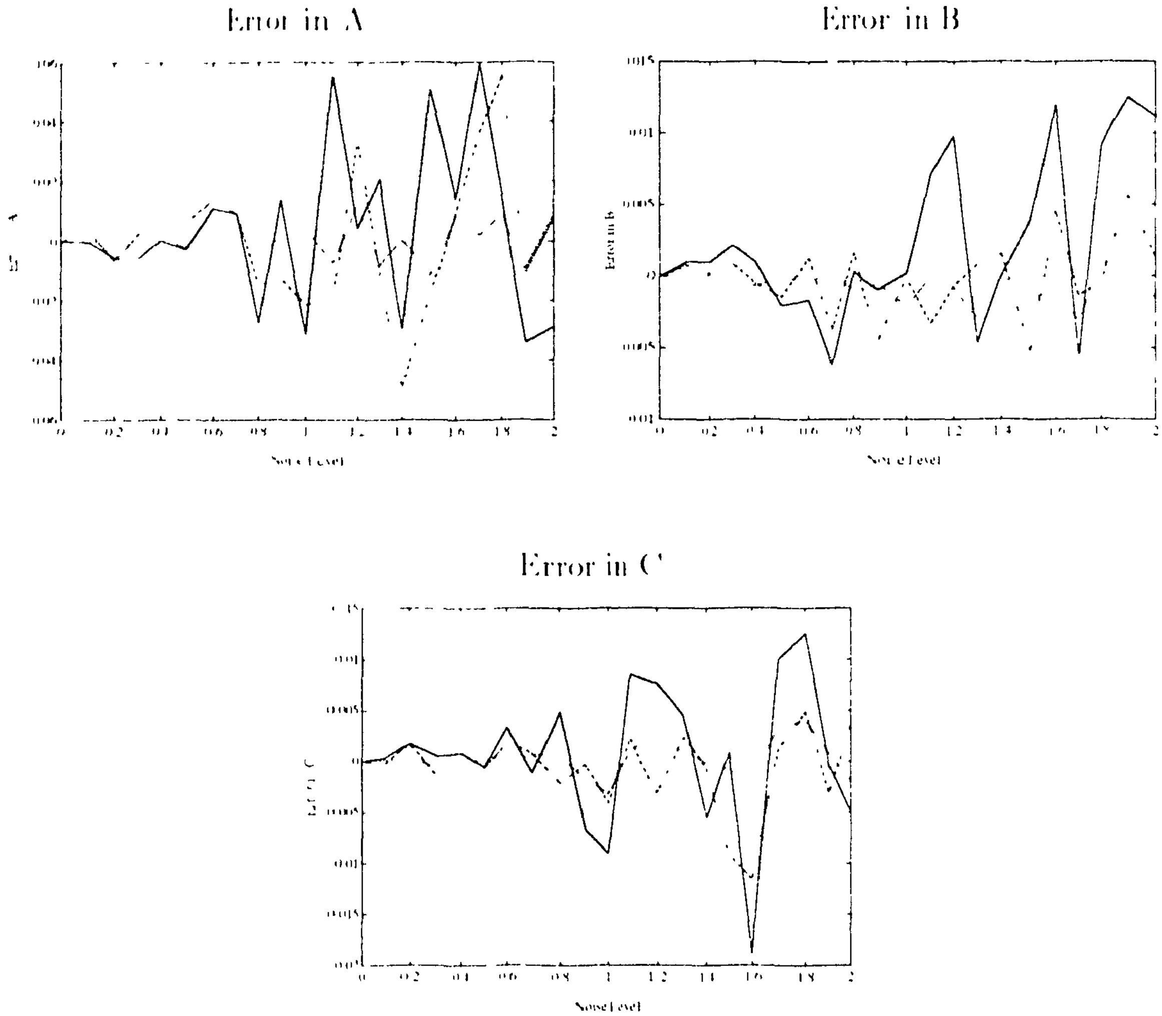
Since the least squares method has a breakdown point of 0%, hence, it does not perform well in the presence of outliers. To test the robustness of the algorithms in the presence of outliers, we add outliers to the synthesized plane. Figure 7.14 shows a plane with 5% and 60% of outliers. Even with this percentage of outliers, it is very

Figure 7.11: Fitting errors vs. Gaussian noise levels (plane, individual experiment)



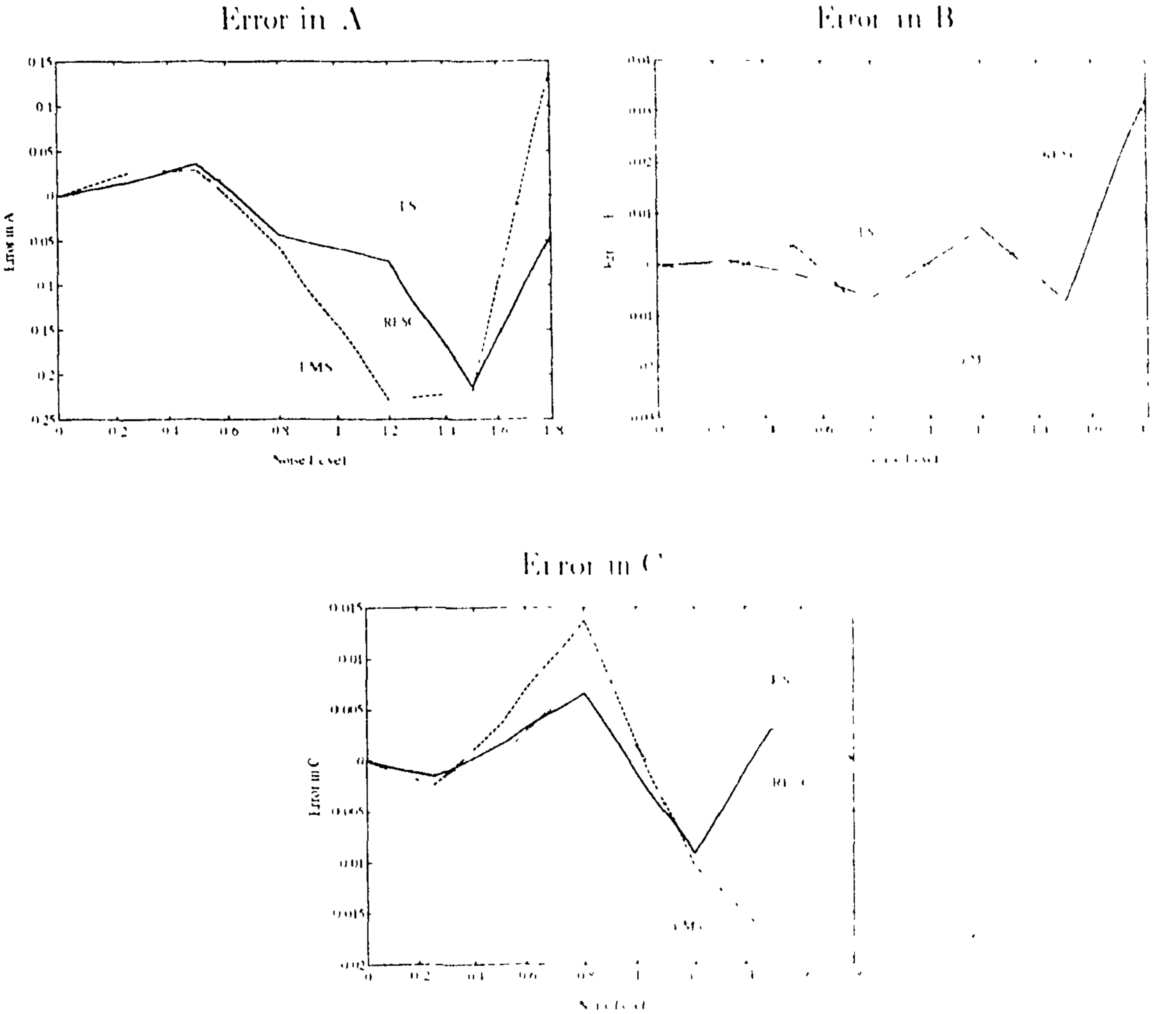
Note: (1) solid line represents errors of the RLSC method; (2) dashed line represents errors of the least median squares method; (3) the dotted line represents errors of the least square method.

Figure 7.12: Fitting errors vs. Gaussian noise levels (synthetic plane, data-average of 30 samples)



Note: (1) solid line represents errors of the RESC method, (2) dashed line represents errors of the least median squares method; (3) the dotted line represents errors of the least squares method;

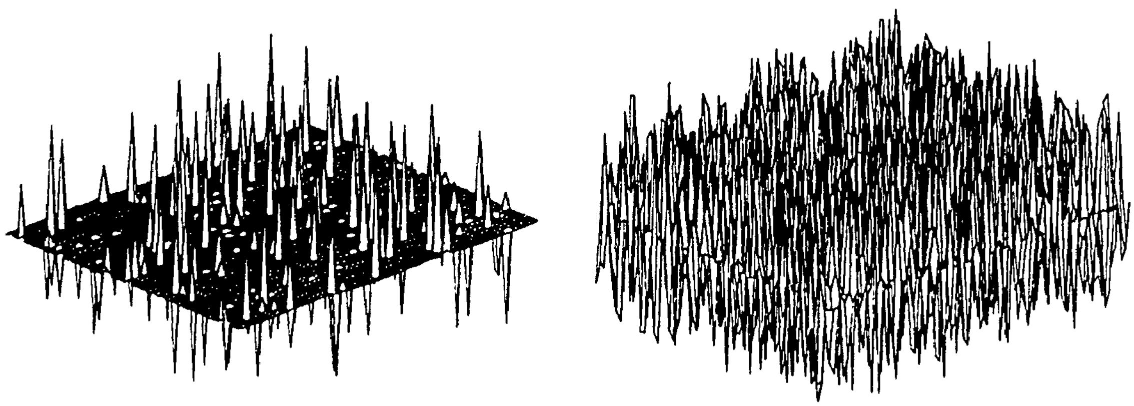
Figure 7.13: Fitting errors vs. Gaussian noise levels (synthetic plane, run average of 20 runs)



Note: (1) solid line represents errors of the RLSC method; (2) dashed line represents errors of the least median squares method; (3) the dotted line represents errors of the least squares method.

Figure 7.14: A plane with outliers

5% outliers 60% outliers



hard to see the original plane from the picture. In Figure 7.15, outliers are added to the synthetic data and the inliers are also perturbed with Gaussian noise with $\sigma = 0.1$. The graphs show clearly that the RESC method is the best when there is a high percentage of outliers. The least squares method can tolerate no outliers. The LMS can tolerate about 50% of outliers. We find to our surprise that RESC can tolerate even 90% outliers in these examples. Figure 7.15 shows only one pass for each outlier percentage. To better understand the algorithm's performance, the results are averaged. For each outlier percentage, we run RESC and LMS 20 times with different seeds. The averages are shown in Figure 7.16 and 7.17. It is clear that RESC can tolerate on the average 80% of outliers, and even 90% in some cases.

Quadratic Primitive

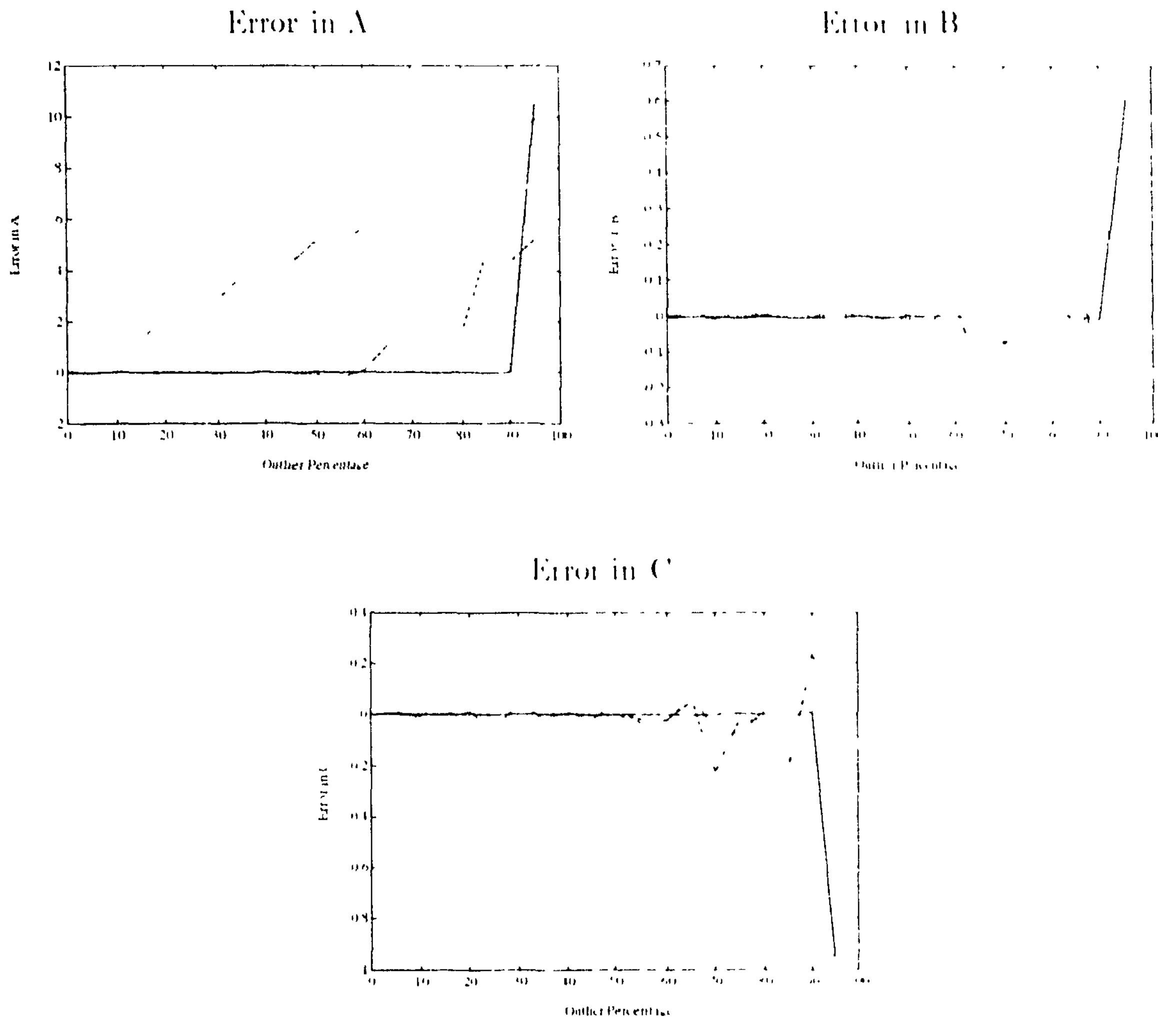
To generate a synthetic range image of a second order primitive, we take an ellipsoid with equation

$$0.01x^2 + 0.01y^2 + 0.02z^2 = 1 \tag{7.4}$$

The normalized eigenvalues derived from this equation (see Appendix C) are:

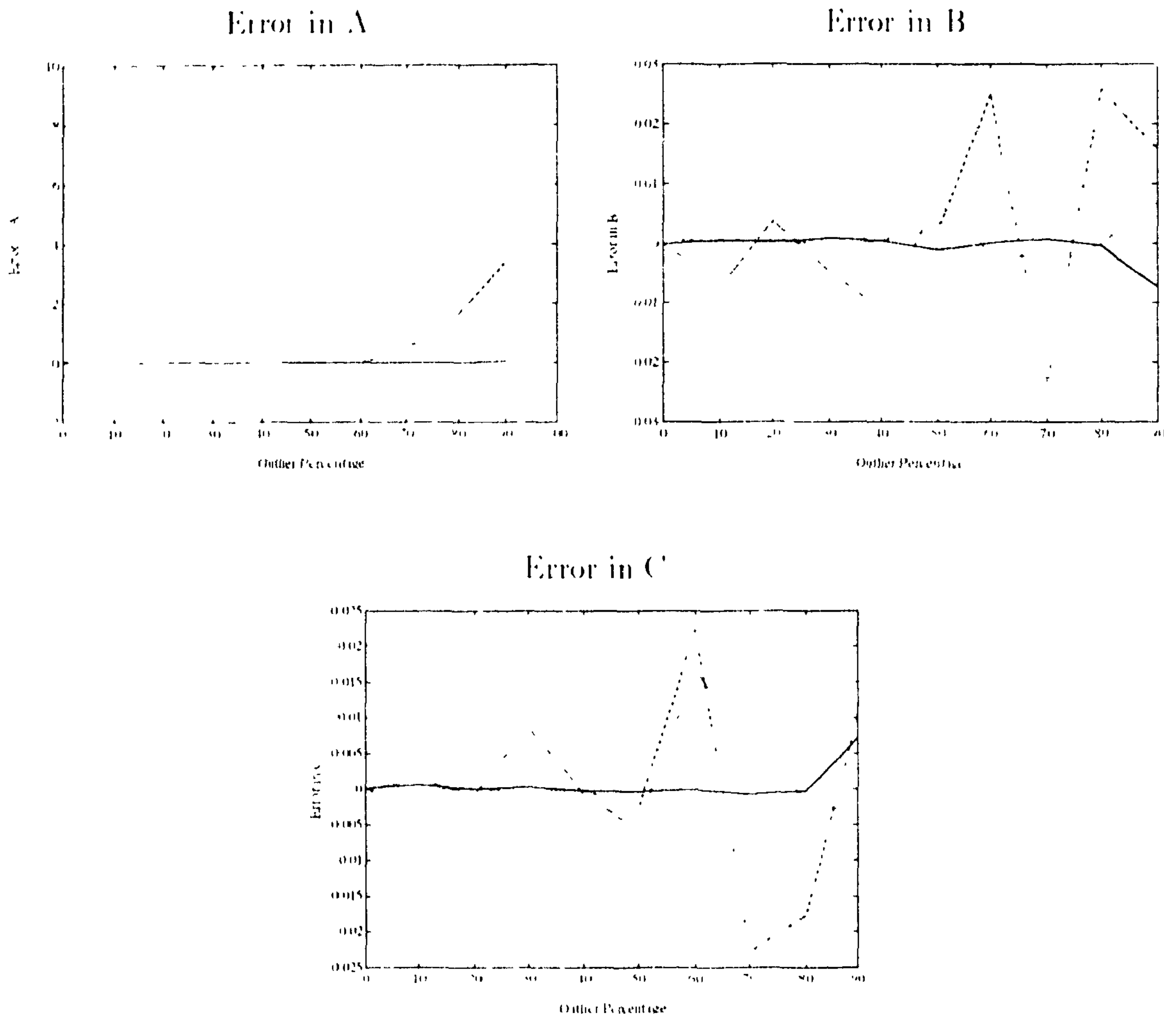
$$\lambda_1 = 1 \qquad \lambda_2 = 0.5 \qquad \lambda_3 = 0.25 \qquad \lambda_4 = -25 \tag{7.5}$$

Figure 7.15: Fitting errors vs. outliers (synthetic plane, individual experiment)



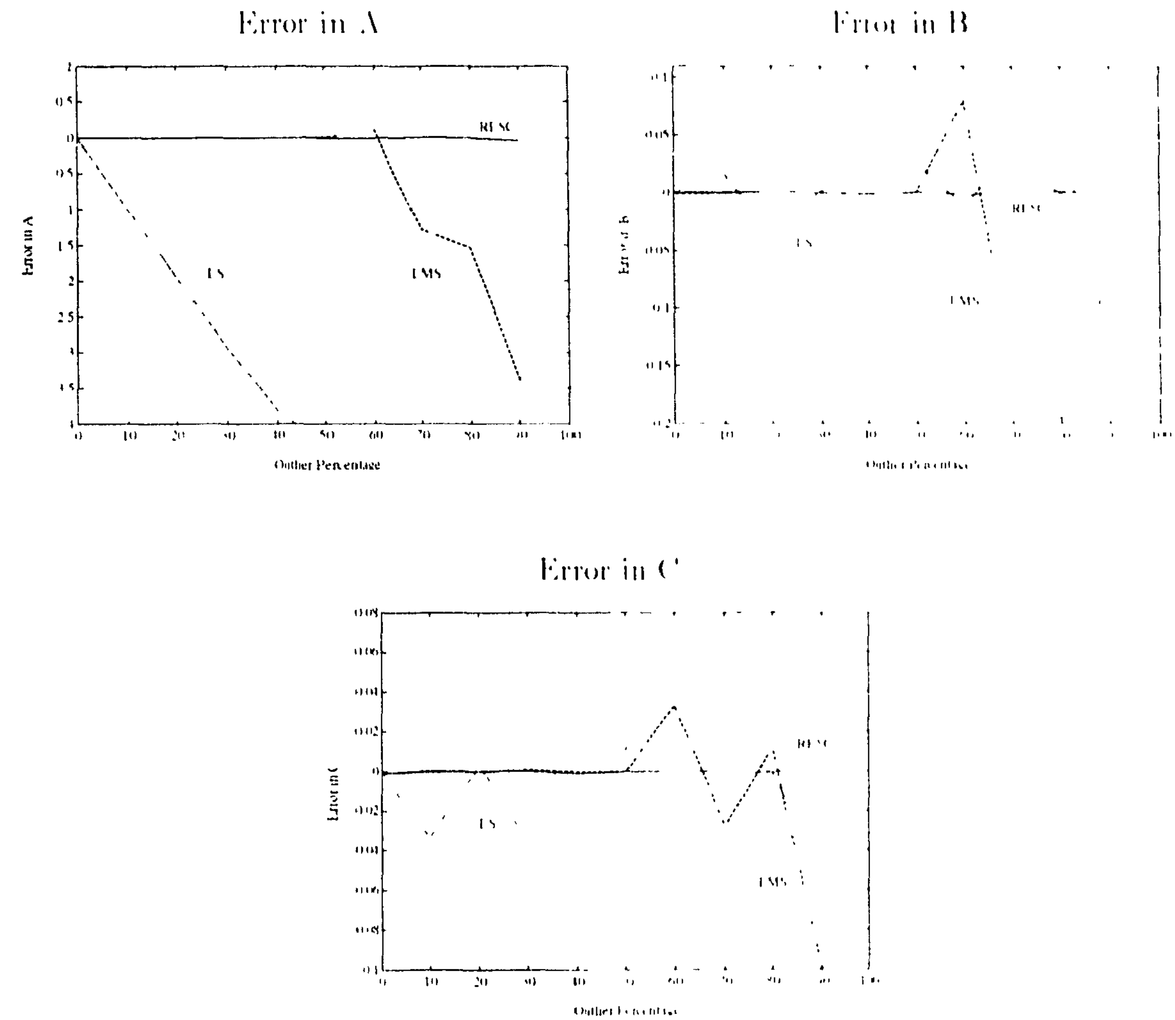
Note: (1) solid line represents errors of the RESC method (2) dashed line represents errors of the least median squares method (3) the dotted line represents errors of the least squares method.

Figure 7.16: Fitting errors vs. outliers (synthetic plane, data-average of 30 samples)



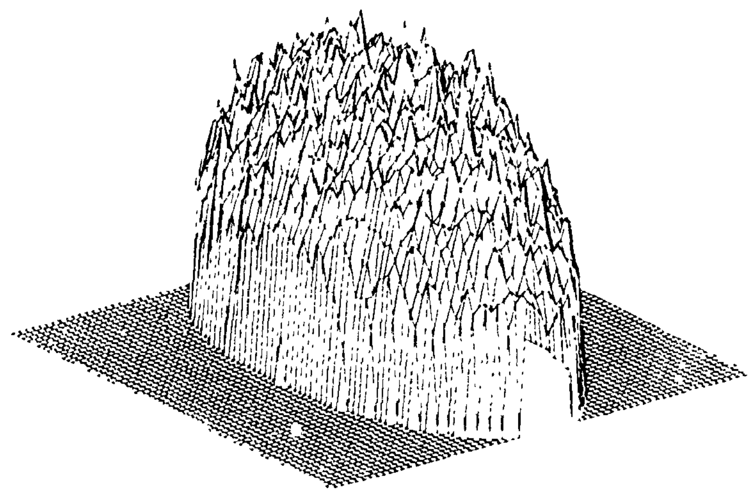
Note: (1) solid line represents errors of the RESC method. (2) dashed line represents errors of the least median squares method; (3) the dotted line represents errors of the least squares method;

Figure 7.17: Fitting errors vs. outliers (synthetic plane, run average of 20 runs)



Note: (1) solid line represents errors of the RESC method, (2) dashed line represents errors of the least median squares method, (3) the dotted line represents errors of the least squares method.

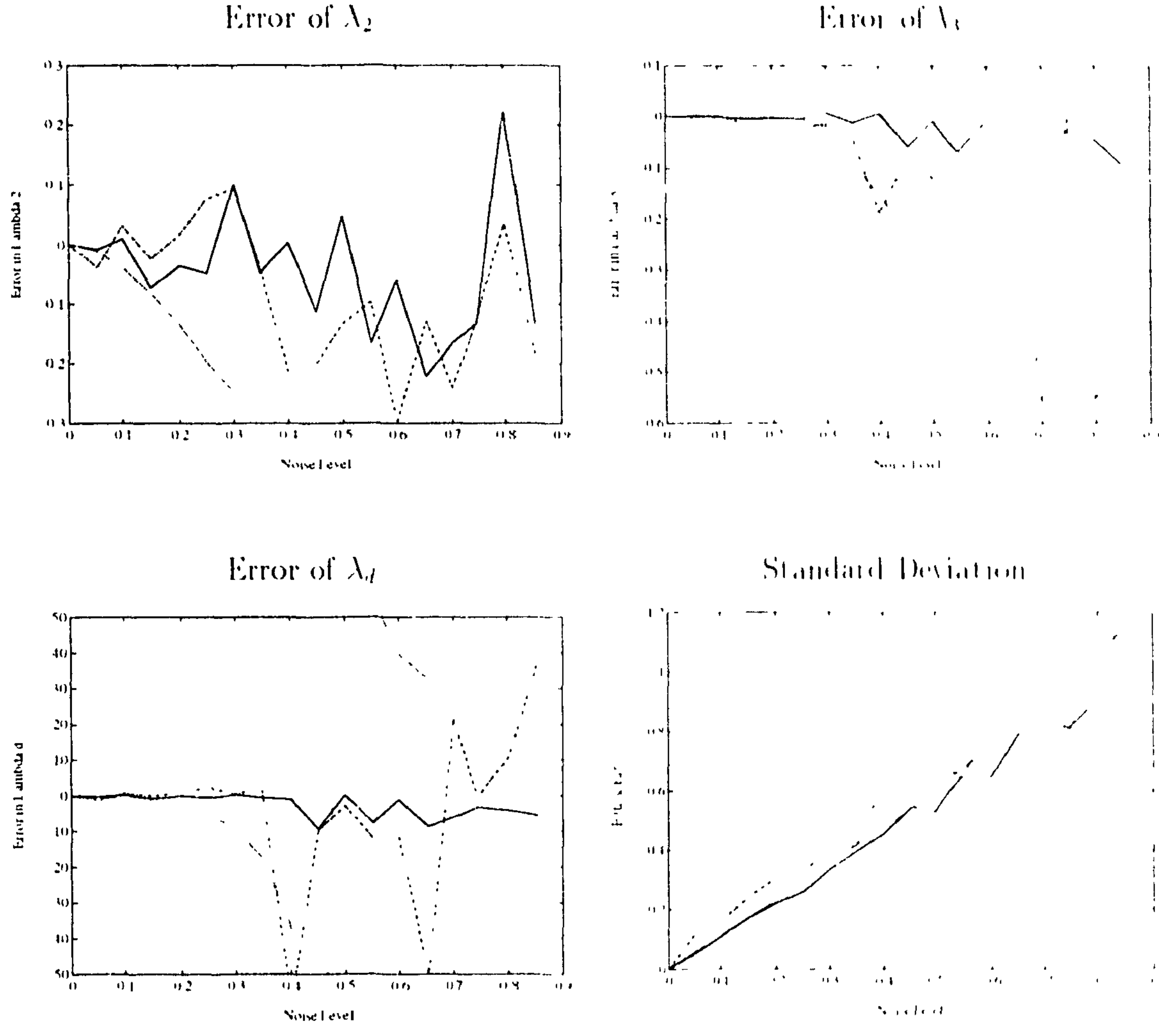
Figure 7.18: An ellipsoid with Gaussian noise ($\sigma = 0.5$)



On a 128×128 grid, z values are generated and the corresponding x and y values are in the range $[-10, 10]$. Given x and y values, z values can be generated from Equation 7.1. For the point outside the ellipsoid, z is assigned to a background value. Gaussian noise is added to z values. Figure 7.18 shows an ellipsoid contaminated with Gaussian noise ($\sigma = 0.5$). In Figure 7.19, which shows fitting errors in the invariants Λ_2, Λ_3 and Λ_4, the least-squares method fits the second-order primitive poorly. This poor fit occurs because the primitive equation involves all the variables implicitly [85]. Compared with the planar cases, the second order primitive is much more difficult to fit correctly. As before, we average the results statistically. Figure 7.20 shows the results of data-average for 30 sets of different data. Figure 7.21 shows the run average of 20 runs.

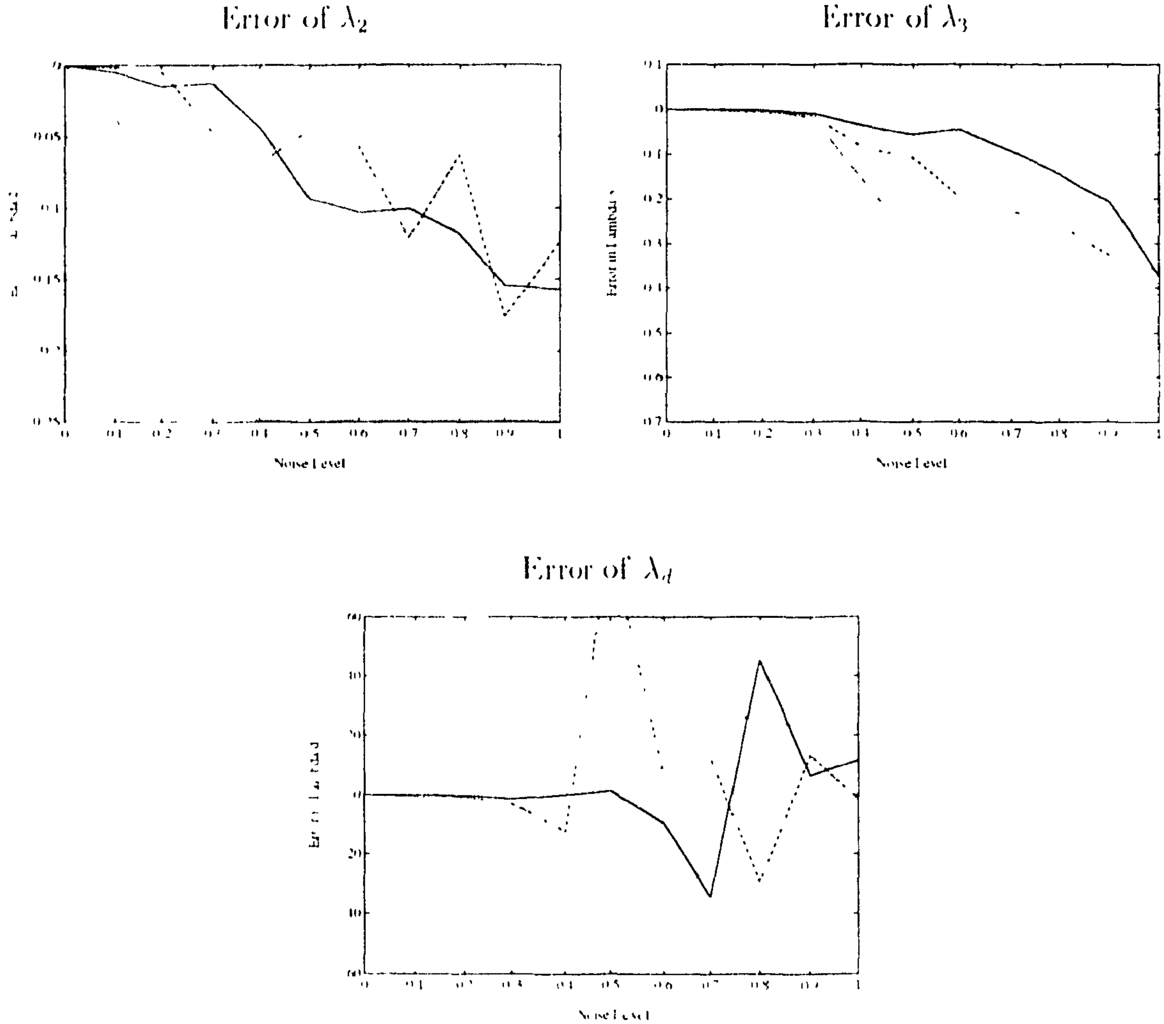
Figure 7.22 shows the ellipsoid contaminated with 5% and 60% of outliers. Outliers are distributed uniformly on the ellipsoid surface range. From Figure 7.19 we can see the errors in the invariants of Λ_2, Λ_3 and Λ_4 respectively. For outlier percentage, 30 sets of different synthetic data are generated. The results are averaged and shown in Figure 7.24. Figure 7.25 shows the run average of 20 runs.

Figure 7.19: Fitting errors vs. Gaussian noise levels (ellipsoid, individual experiment)



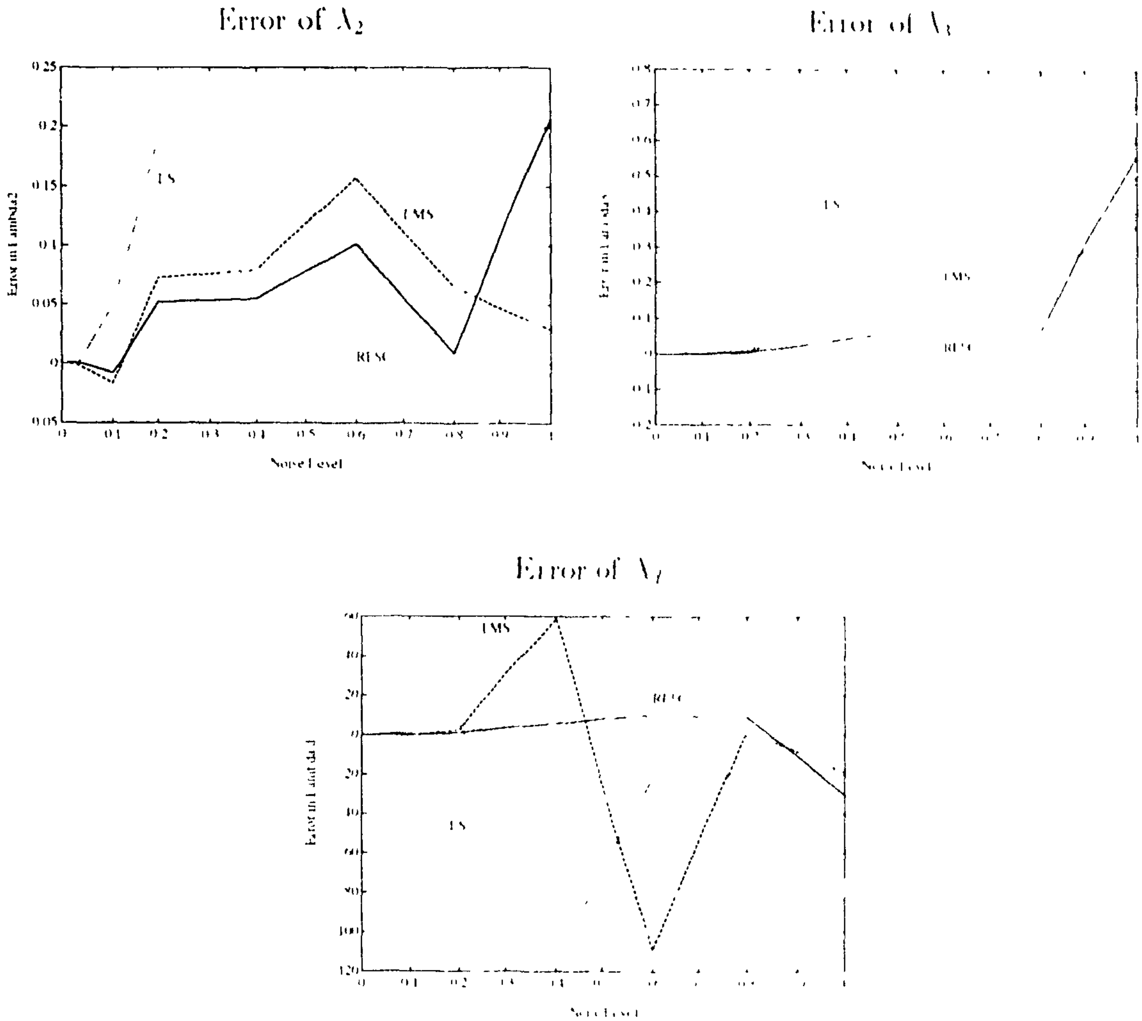
Note: (1) solid line represents errors of the RESC method (2) dashed line represents errors of the least median squares method, (3) the dotted line represents errors of the least squares method;

Figure 7.20: Fitting errors vs. Gaussian noise levels (ellipsoid, data-average of 30 samples)



Note: (1) solid line represents errors of the RESC method. (2) dashed line represents errors of the least median squares method; (3) the dotted line represents errors of the least squares method;

Figure 7.21: Fitting errors vs. Gaussian noise levels (ellipsoid, run average of 20 runs)



Note: (1) solid line represents errors of the RESC method; (2) dashed line represents errors of the least median squares method; (3) the dotted line represents errors of the least squares method;

7.3.2 Analysis of Experimental Results

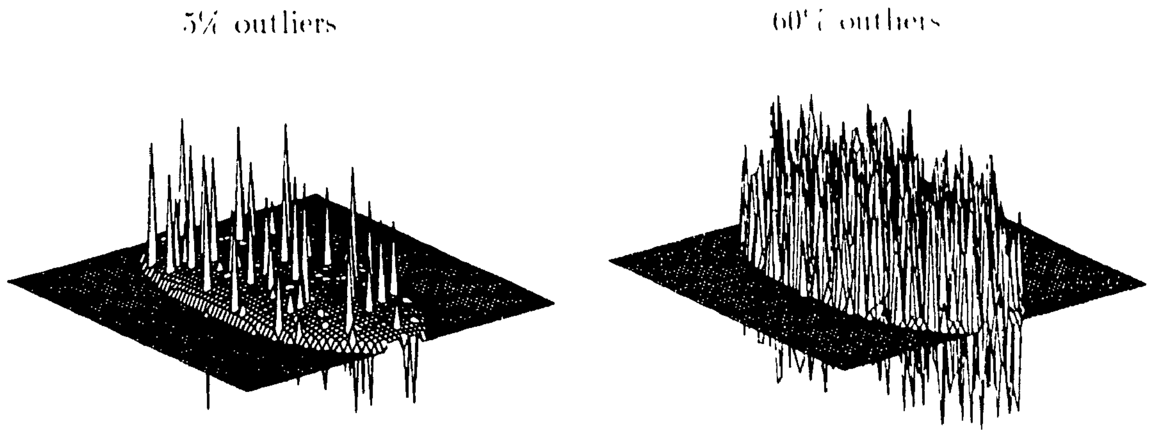
The least-squares method fits the second-order primitive poorly. This poor fit occurs because the primitive equation involves all the variables implicitly [85]. The least squares minimizes the so-called *algebraic distance*, or difference between the two sides of the fitting equation, rather than the geometric residual. This is not a drawback of the least squares method itself, but of the way it is used.

It is impossible to use the linear least squares to minimize the geometric distance. Taubin [78] derives expressions for approximate orthogonal distance from curves and surfaces given in implicit form. The distance is in the direction perpendicular to the surface normal, whereas the error of the real range image is mainly in the z direction. The minimization of the approximate mean square distance is a nonlinear least squares problem, although in certain cases it reduces to a generalized eigenvector fit. In the general case one has to use an iterative Levenberg-Marquardt algorithm, involving extensive computation. The method investigated in [78] may fail if there are outliers. The LMS method uses the median residual to represent the fitting for the whole region. There is no particular reason to select the median residual as a criterion. Why not at the 70% or 30% quantile? We do not know which position divides inlier part from the outlier part by the LMS method. In outlier-free cases, the least median is a weak criterion. The RESC method solves this problem by a histogram method to correctly estimate the whole inlier part, and the optimization is based on the whole inlier part. Figure 7.24 and Figure 7.25 prove that RESC is successful not only in handling outliers but also in handling Gaussian noise.

The RESC method is not fast compared with the least-squares method, but it is robust with respect to outliers. The least median squares method and the RESC method work well in the presence of Gaussian noise, although the estimated value is somewhat inaccurate for the following reasons:

1. LMS and RESC take a small number of points from the sample region to

Figure 7.22: An ellipsoid with outliers



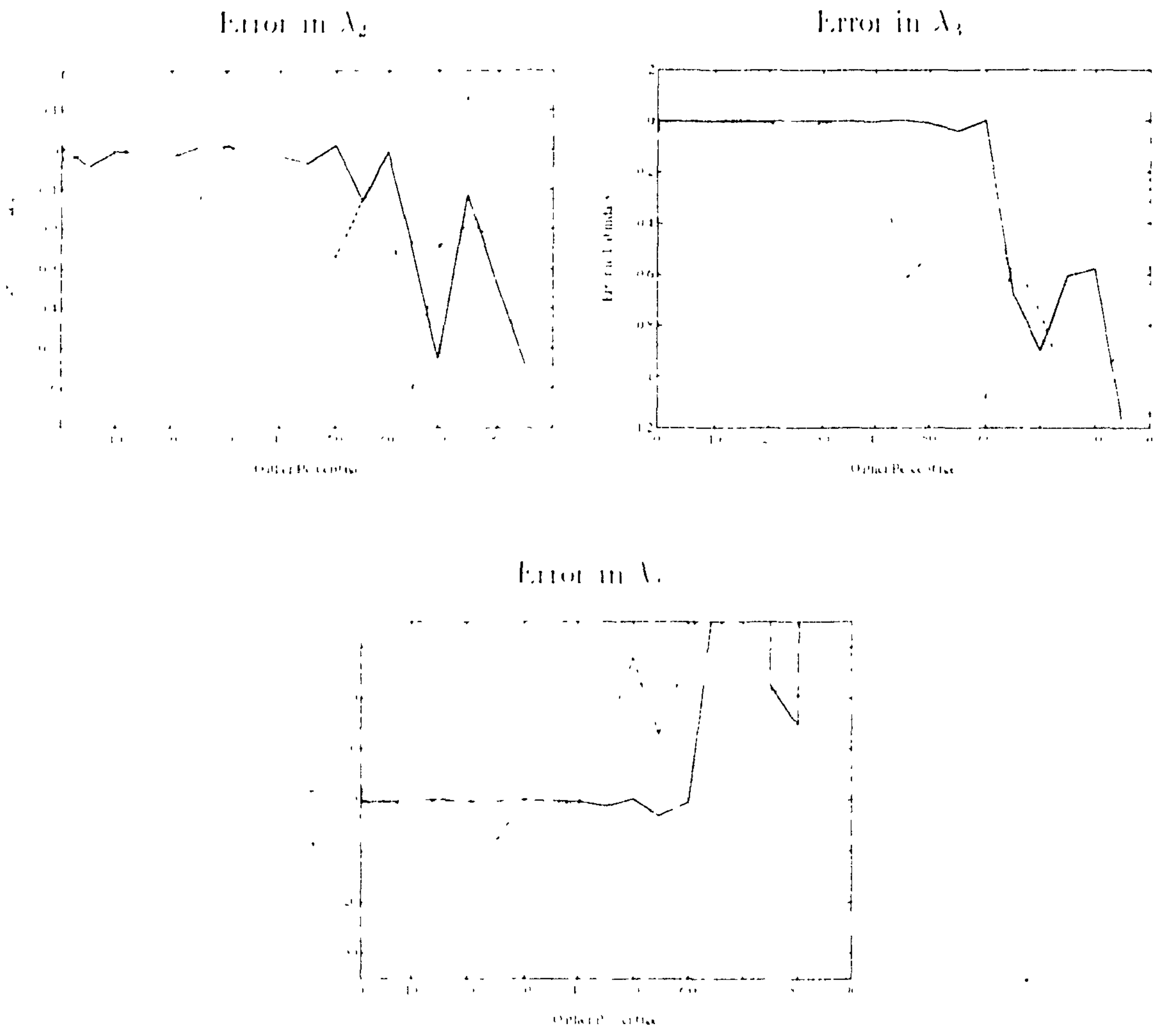
solve the equation (3 for planar primitive and 9 for quadratic primitive). The accuracy of the results depends on the particular points chosen. The estimated value may vary slightly from one set of points to another.

2. These methods, being stochastic searches, do not find the best solution deterministically. Rather they converge asymptotically to the optimal solution in a probabilistic sense.

Figure 7.23 and Figure 7.25 show the fitting results vs. the percentage of outliers. The least squares method tolerates no outliers. LMS tolerates about 40% of outliers and RESC 60% of outliers. Compared with the planar surface fitting case, the curved surfaces are much more difficult to fit accurately. The breakdown point of the method is lower than in the case of a planar primitive.

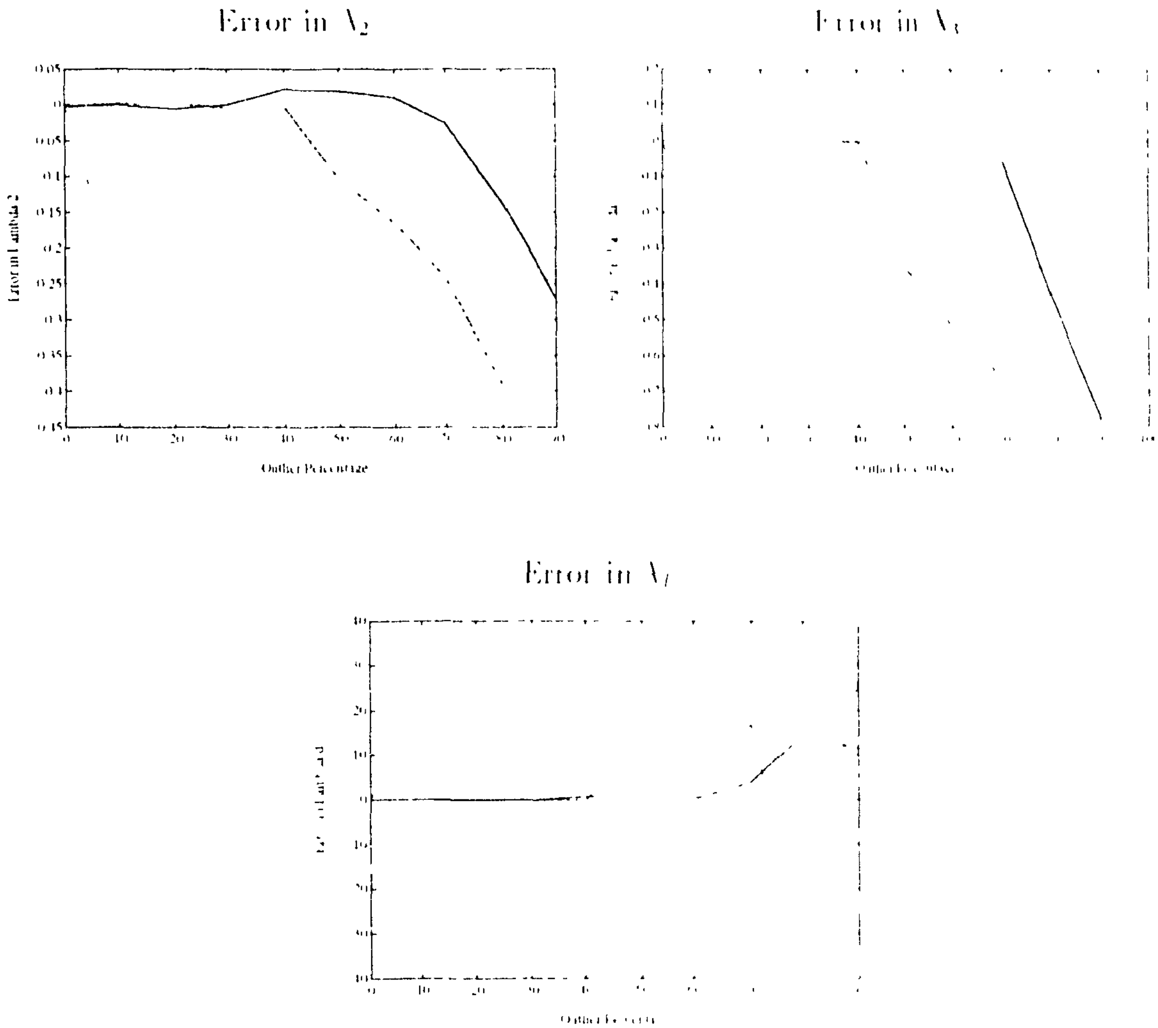
We can see that 80% breakdown point can only be achieved at the low level of noises. With increase of the noise level, the breakdown point is decreasing. Figure 7.26 shows the influence of noise on the breakdown point of the RESC. The higher the noise level, the smaller the breakdown point. When the noise level is high, accurate fitting is difficult even with no outliers. Clearly, the resistance of an estimator to outliers can be influenced by noise level. What is the highest breakdown

Figure 7.23: Fitting errors vs. outliers (ellipsoid, individual experiment)



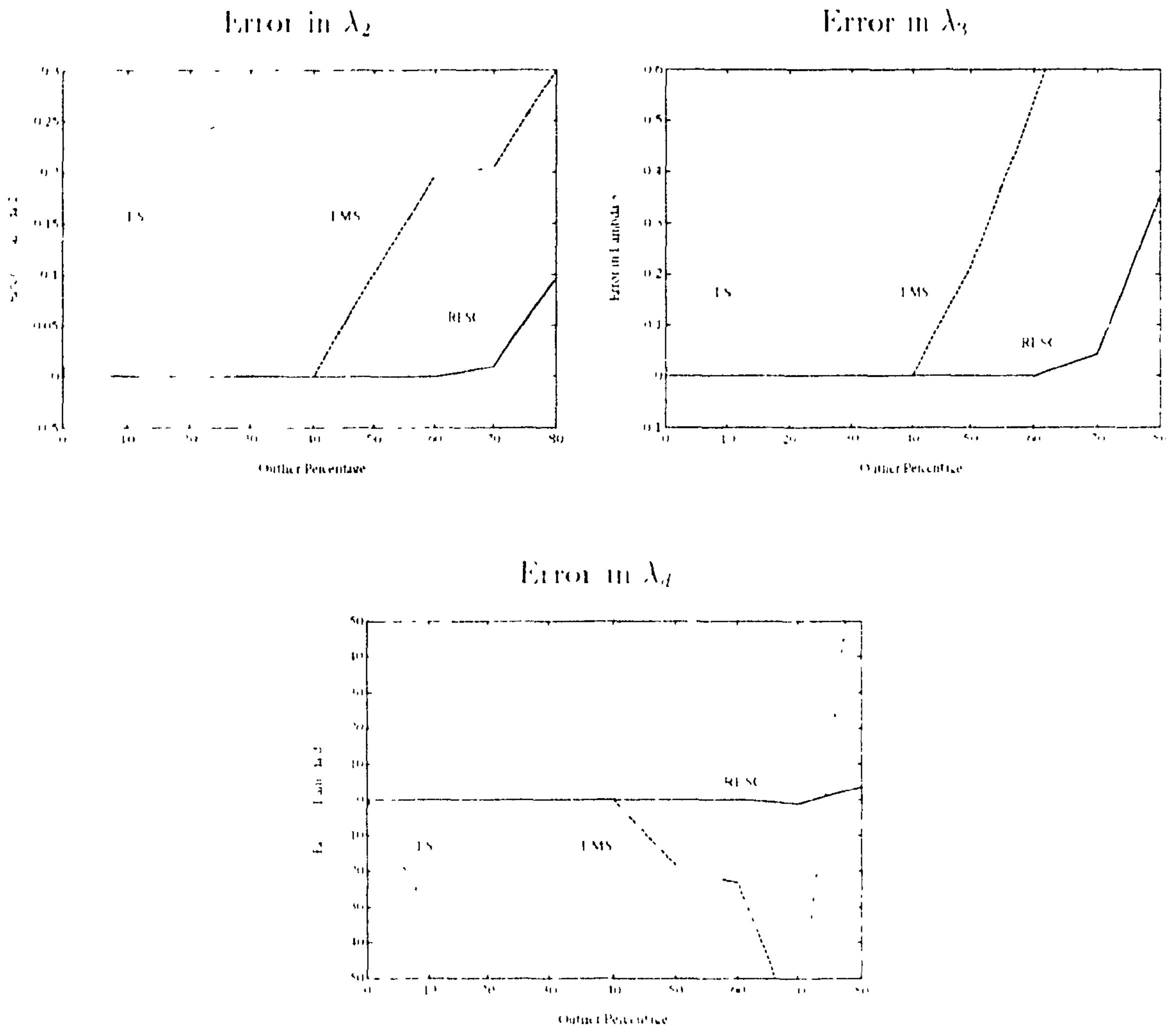
Note: (1) solid line represents errors of the RESC method; (2) dashed line represents errors of the least median squares method; (3) the dotted line represents errors of the least squares method;

Figure 7.21: Fitting errors vs. outliers (ellipsoid, data average of 30 samples)



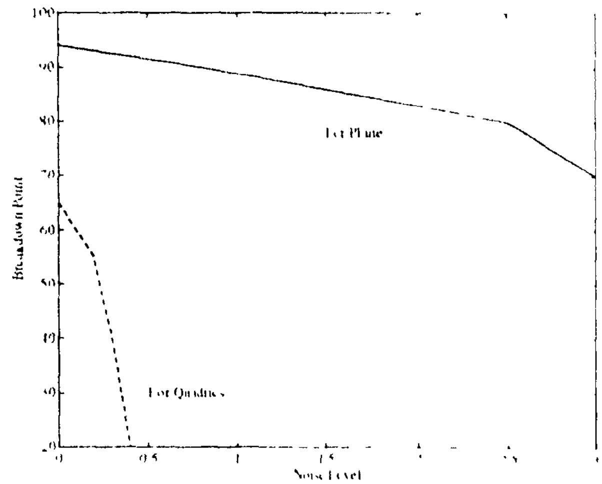
Note: (1) solid line represents errors of the RLSC method (2) dashed line represents errors of the least median squares method (3) the dotted line represents errors of the least squares method

Figure 7.25: Fitting errors vs. outliers (ellipsoid, run-average of 20 runs)



Note. (1) solid line represents errors of the RESC method; (2) dashed line represents errors of the least median squares method; (3) the dotted line represents errors of the least squares method;

Figure 7.26: Breakdown points vs. Gaussian noise level



point a robust estimator can achieve? We have observed that RESC method achieves 91% breakdown point for planar primitive fitting. Rousseeuw and Leroy [69] prove that 50% breakdown point is the best achievable for robust estimation method. The proof of the theorem shows that they require the estimation method to produce a unique solution. When there are 50% or more outliers, the estimation method may have multiple solutions. Our applications do not require the solution to be unique. In case of multiple solutions, the RESC method will choose the one which has the best residual consensus.

7.3.3 Real Range Data Experiments

In our 3D range image experiments, primitives are planar and general quadratic surfaces. The primitive equations are

$$F(x,y,z) = Ax^2 + By^2 + Cz^2 + Dxy + Eyz + Fxz + Gx + Hy + Iz + J \quad (7.6)$$

for the second order primitive and

$$z = Ax + By + C \quad (7.7)$$

Figure 7.27: The grip

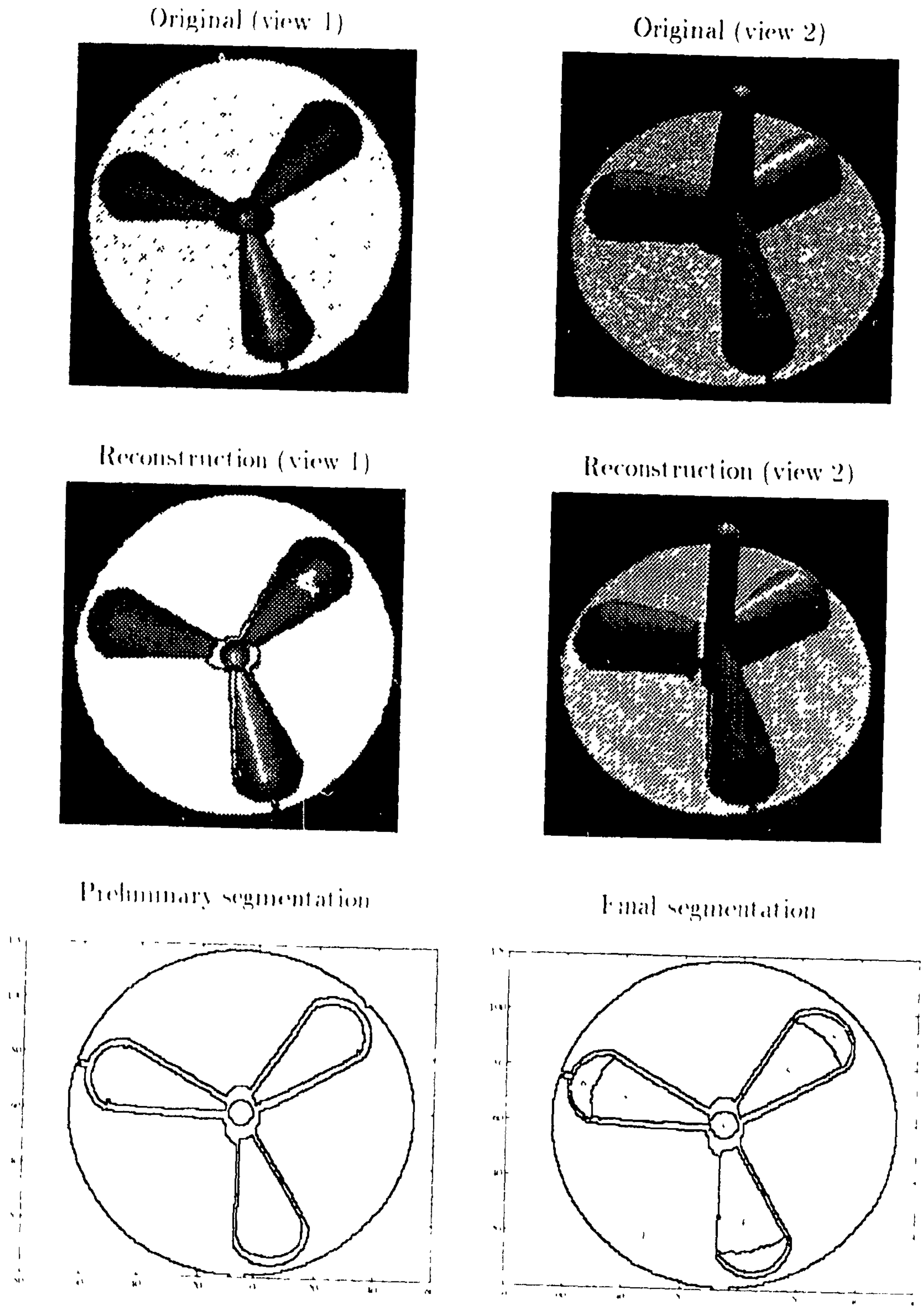


Table 7.2: Fitting data for the grip

L	N	E	Equation Parameters	Λ	Rotation Matrix	[illegible]
1	32892	0.07	12.139 0.004103 0.0016			
2	2259	0.06	-0.000038 -0.000234 0.000216	0.644	0.4496 0.3346 0.9301	294
			-0.000127 0.000118 -0.000042	0.042	0.7917 0.5232 0.3155	168
			0.003630 0.028161 0.015497	5797	0.5923 0.7838 0.1866	110
3	325	0.10	0.000363 0.000361 -0.000271	0.968	0.9723 0.2012 0.1189	98
			0.000009 -0.000037 0.000154	0.752	0.2060 0.9781 0.0292	85
			0.007354 -0.058236 0.049796	1364	0.1101 0.0529 0.9925	[illegible]
4	2270	0.07	0.018542 0.001309 0.012600	0.605	0.9125 0.2246 0.21?6	96
			0.009004 0.001637 -0.004504	0.003	0.2137 0.1648 0.96?9	33?
			-0.350275 -0.073144 -0.237327	564	0.2570 0.9604 0.10?3	18
5	2287	0.06	-0.000159 -0.000167 0.000266	0.948	0.6114 0.3965 0.6849	?50
			0.000280 -0.000116 -0.000106	0.000	0.7108 0.01?7 0.6?45	18?
			-0.014752 0.024174 0.021034	4	0.2784 0.9179 0.28?8	1?9
6	634	0.06	-0.000145 -0.000153 0.000101	0.922	0.3870 0.9211 0.01?8	44
			-0.000009 -0.000011 0.000003	0.642	0.9181 0.3806 0.1103	?0
			-0.012161 0.021181 0.001882	444	0.08?2 0.08?0 0.9930	[illegible]
7	613	0.07	-0.003288 -0.003245 0.002422	0.963	0.8688 0.4660 0.16?5	5
			-0.000108 -0.000013 0.000301	0.717	0.1754 0.8?96 0.0184	18
			0.029881 0.145287 0.065565	462	0.138? 0.0956 0.98??	44
8	609	0.07	0.000128 -0.000126 0.000086	0.980	0.9100 0.4056 0.08?8	?4
			0.000003 0.000010 0.000007	0.666	0.1137 0.9020 0.1?44	[illegible]
			0.005710 0.021729 0.001478	244	0.0271 0.117? 0.98?6	44

Figure 7.28: Harris cup

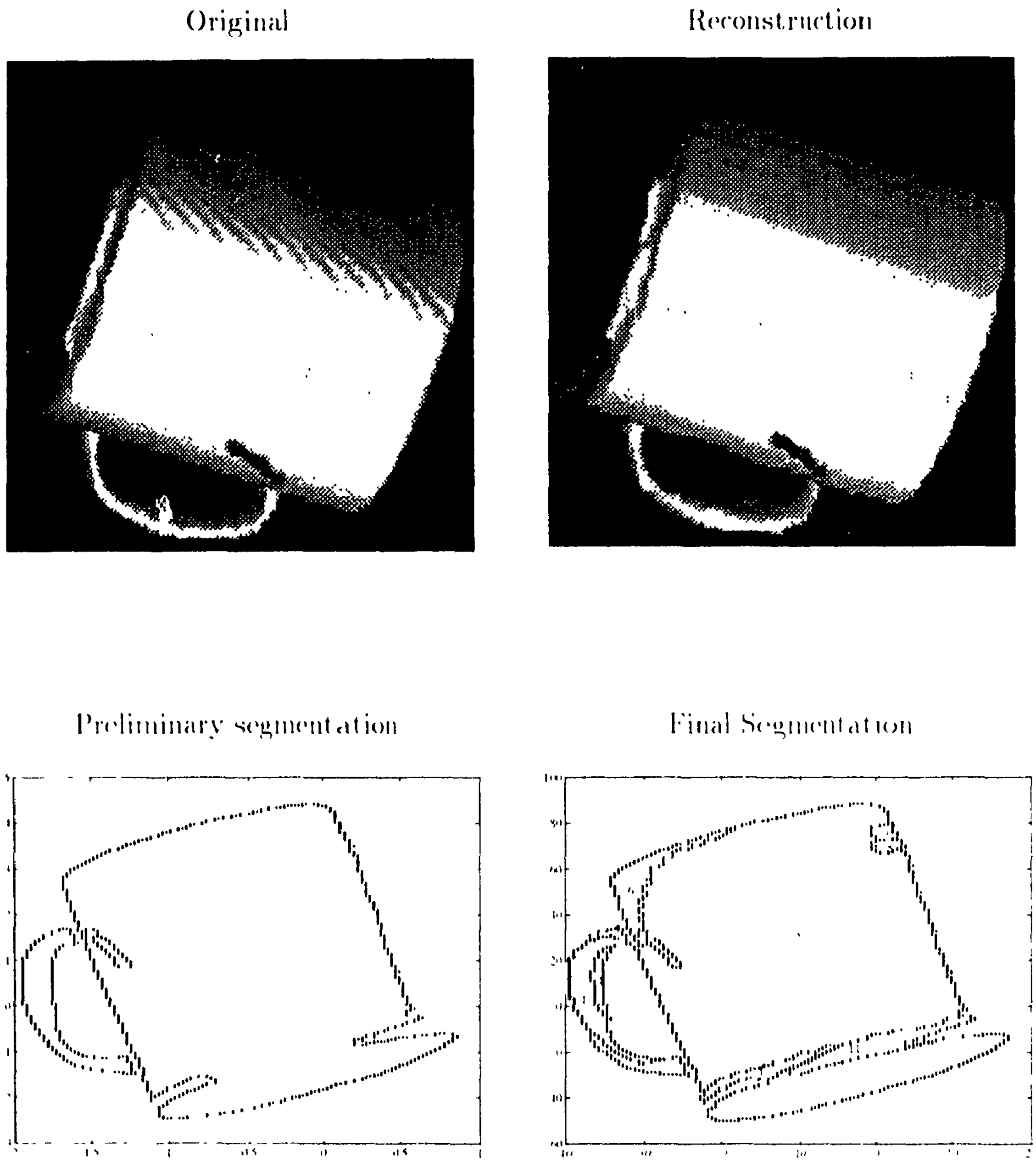


Table 7.3: Fitting data for the Harris cup

L	N	E	Equation Parameters	λ	Rotation Matrix	It.
1	410	0.21	-0.000034 0.000011 -0.000082	0.192	0.0239 0.9977 0.0630	344
			0.000011 0.000270 0.000016	-0.605	0.5794 0.0651 0.8124	43
			-0.020775 -0.000868 0.019588	-13630	0.8147 0.0171 0.5796	49
2	8152	0.10	-0.000267 -0.000006 -0.000122	0.156	0.9899 0.0525 0.1319	197
			-0.000068 0.000031 0.000025	0.003	0.1232 0.1435 0.9820	3807
			-0.006162 0.001947 0.023427	14571	0.0701 0.9883 0.1356	[illegible]
3	137	0.15	0.000032 0.000134 0.000012	0.438	0.6483 0.3615 0.6701	40
			-0.000186 0.000022 0.000307	0.735	0.6489 0.7227 0.2379	54
			-0.007333 0.022420 0.012546	1666	0.3983 0.5891 0.7034	63
4	71	0.09	-0.000304 0.000053 -0.000064	0.334	0.9781 0.1003 0.1824	[illegible]
			0.000115 0.000172 0.000112	0.353	0.1093 0.4980 0.8602	[illegible]
			-0.038588 0.001087 0.010320	747	0.1771 0.8613 0.4762	54
5	35	0.15	-0.000070 -0.000002 -0.000071	0.349	0.4786 0.8767 0.0495	49.3
			0.000054 0.000151 -0.000059	0.331	0.4851 0.2471 0.8469	171
			-0.017109 0.006044 0.017180	2175	0.7317 0.4293 0.5294	20
6	323	0.08	-0.000286 -0.000009 0.000110	0.391	0.9960 [illegible] [illegible]	[illegible]
			-0.000045 0.000036 -0.000018	0.011	0.0772 0.1762 0.9814	[illegible]
			-0.010126 0.002106 0.021256	9165	0.0139 0.9839 [illegible]	[illegible]

Table 7.1: Fitting data for the Harris cup (continued)

L	N	E	Equation Parameters	λ	Rotation Matrix	Tr.
7	1300	0.29	0.000297 0.000167 0.000041	-0.026	0.7902 0.4759 -0.3863	-82
			-0.000499 -0.000027 0.000102	-0.148	-0.6061 0.7000 -0.3776	-89
			0.004802 -0 008739 0.013188	336	0.0907 0.5325 0.8446	89
8	37	0.06	0.000075 0.000006 -0.000052	0.176	0.5190 0.8523 -0.0644	111
			0.000046 -0.000146 -0.000037	-0.391	0.5249 -0.2584 0.8110	75
			0.019836 0.005069 0.014388	-1972	0.6746 -0.4547 -0.5815	-6
9	34	0.10	0.000155 -0.000007 -0.000053	0.569	-0.8464 0.5285 0.0654	65
			0.000064 -0.000179 -0.000038	-0.355	0.3633 -0.4833 -0.7965	68
			0.024529 0.005202 0.014582	128	-0.3893 -0.6980 0.6010	0
10	128	0.09	0 000198 -0.000268 0.000085	-0.130	-0.3601 -0.3592 0.8610	38
			0.000375 0.000089 0.000271	-0.911	0.9102 -0.3378 0.2397	-55
			0.022836 0.049392 0.004367	2631	0.2048 0.8699 0.4486	64
11	272	0.15	0.000051 0.000491 -0.000129	0.466	0.5253 0.1679 0.8342	4
			-0.000875 0.000051 0.000082	-0.294	0.8495 0.1595 0.5028	4
			0.014527 0.007926 0 025682	355	0.0486 0.9728 -0.2265	100

Figure 7.29: Bigwye

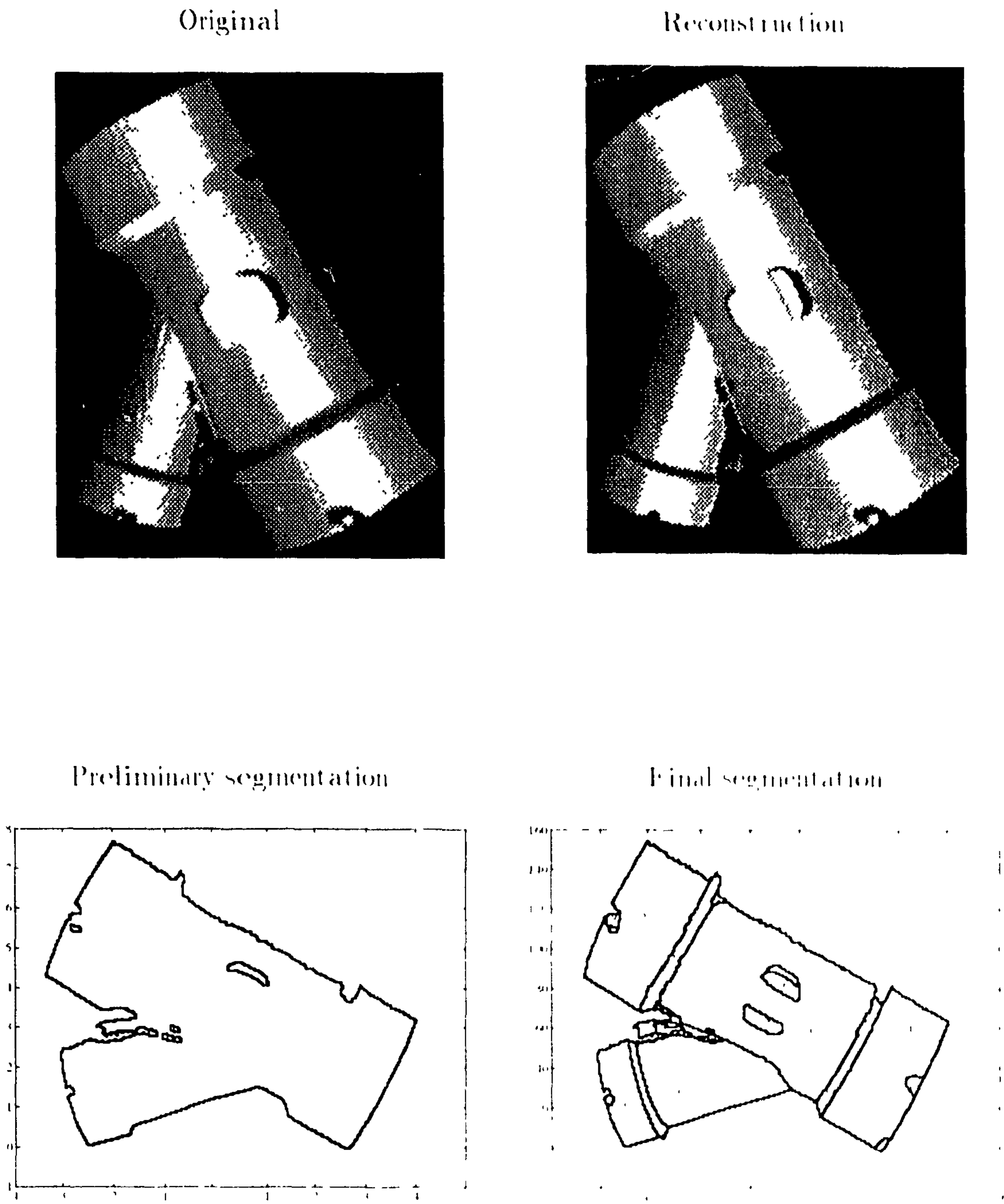


Table 7.5: Fitting data for the bigwye

L	N	E	Equation Parameters	λ	Rotation Matrix	Tr.
1	7565	0.17	0.000038 0.000077 -0.000120	0.930	-0.1121 -0.5599 -0.8209	887
			-0.000105 0.000010 0.000007	0.007	-0.2422 -0.7858 0.5691	-530
			0.009973 0.012553 0.016966	-8997	-0.9637 0.2627 -0.0475	118
2	3740	0.08	0.000037 0.000065 -0.000111	0.856	-0.1660 -0.5737 0.8020	-3
			0.000094 -0.000008 -0.000005	0.016	0.2287 0.7687 0.5973	88
			0.008398 0.011719 0.017337	-1213	-0.9592 0.2826 0.0036	73
3	3096	0.09	0.000021 0.000131 -0.000171	0.742	-0.1162 0.3479 0.9303	395
			0.000097 0.000046 0.000008	0.009	0.4703 -0.8057 0.3600	89
			0.008088 0.012702 0.023145	-3002	-0.8748 -0.4791 0.0700	15
4	3636	0.09	-0.000037 -0.000078 -0.000125	0.899	-0.0409 0.5617 -0.8263	608
			0.000101 -0.000010 0.000008	0.013	-0.1502 -0.8142 0.5608	-330
			0.010019 0.012389 0.017866	7501	-0.9878 0.1470 0.0511	105
5	155	0.05	0.000000 0.000002 -0.000109	0.016	0.1478 0.0143 -0.9890	-105
			0.000002 0.000017 0.000033	-0.017	0.0759 0.9971 0.0000	-153
			0.001596 -0.003205 0.020930	-391	-0.9861 0.0751 -0.1482	0
6	189	0.07	0.000008 0.000020 -0.000168	0.071	0.2989 -0.0135 -0.9542	201
			0.000023 0.000061 0.000143	-0.051	0.1849 0.9801 0.0718	117
			0.004465 0.008631 0.025931	1033	0.9362 -0.1979 -0.2904	-42

Table 7.6: Fitting data for the bigwye (continued)

L	N	E	Equation Parameters	λ	Rotation Matrix	[illegible]
7	512	0.07	0.000148 -0.000012 -0.000125	0.189	0.8858 0.1255 0.1852	9
			-0.000270 0.000031 0.000017	0.598	0.1624 0.7720 0.1361	92
			0.023691 0.007335 0.020652	1903	0.042 [illegible] 1721 0.8805	94
8	58	0.10	-0.000012 -0.000015 0.000043	0.131	0.1636 0.9817 0.0603	[illegible]99
			-0.000032 0.000186 -0.000029	0.728	0.5893 0.0185 0.8061	[illegible]
			-0.019618 0.005061 0.003178	39182	0.7911 0.1675 0.5883	[illegible]
10	1320	0.07	-0.000016 0.000125 -0.000181	0.719	[illegible]	[illegible]
			0.000091 0.000011 0.000028	0.019	0.3231 [illegible]	61
			-0.007392 0.010503 0.024509	970	0.9462 0.3044 0.1108	[illegible]
11	348	0.08	-0.000007 -0.000052 -0.000115	0.829	0.1903 0.5518 0.8099	1
			-0.000135 0.000020 -0.000003	0.355	0.3363 0.7383 0.5817	99
			0.013608 0.008953 0.015565	386	0.9223 0.3836 0.0161	[illegible]
13	75	0.05	0.000001 0.000000 0.000093	0.002	0.1259 0.2558 0.9585	[illegible]
			0.000000 0.000008 0.000024	0.029	0.0389 0.9667 0.2529	5344
			0.000988 -0.002723 0.019394	64885	0.9913 0.0055 0.1317	50

Figure 7.30: Meca17

Original | Reconstruction

Preliminary Segmentation | Final Segmentation

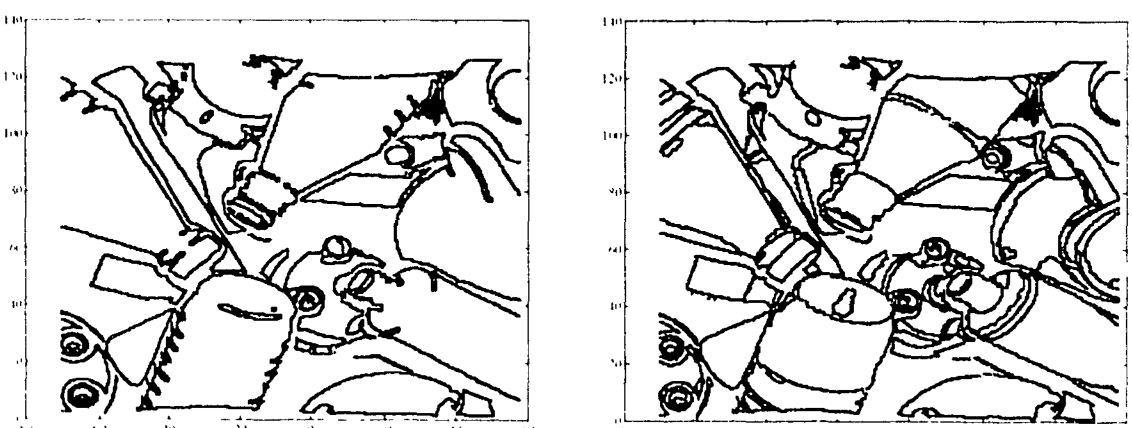


Figure 7.31: The Tube

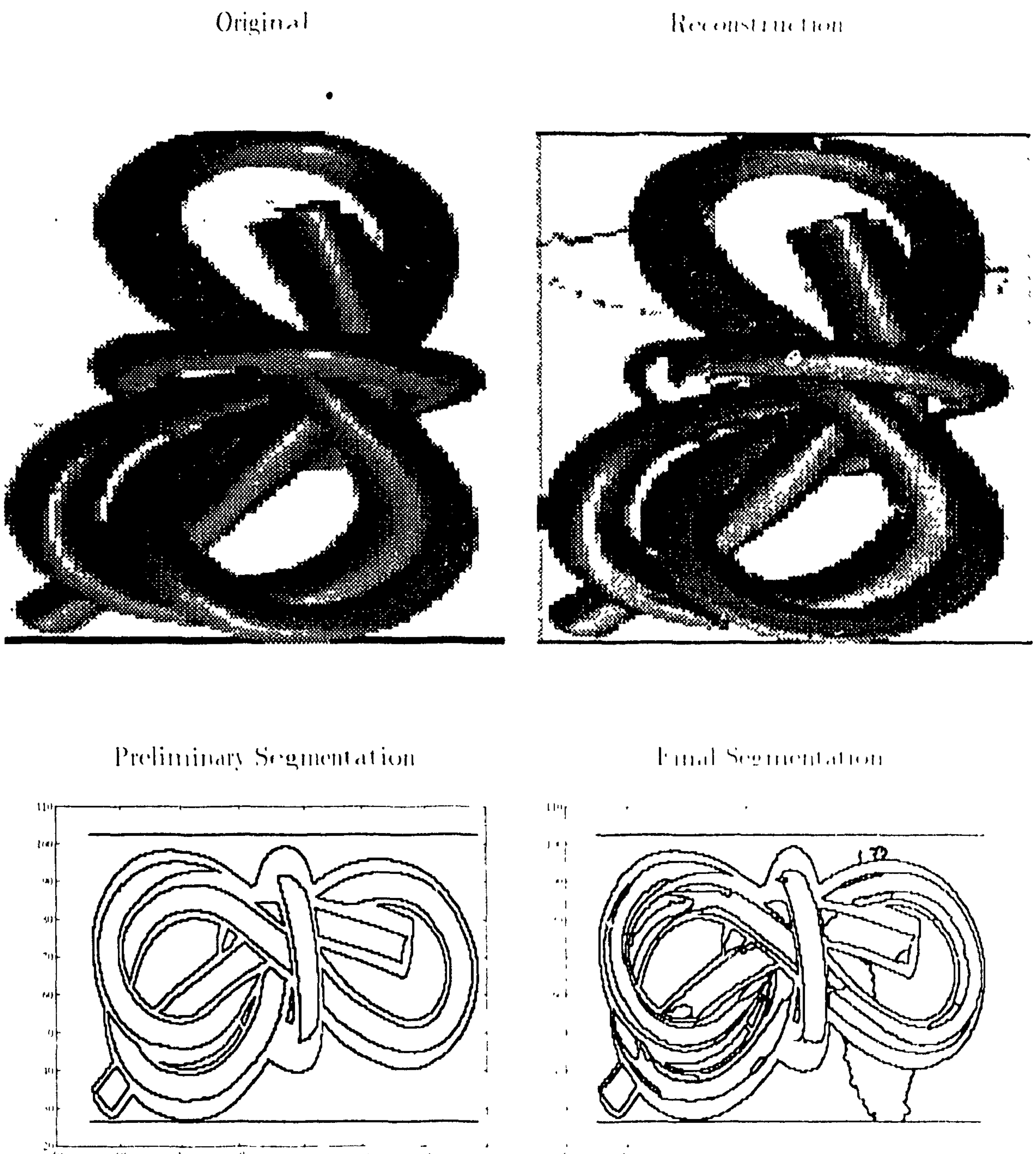
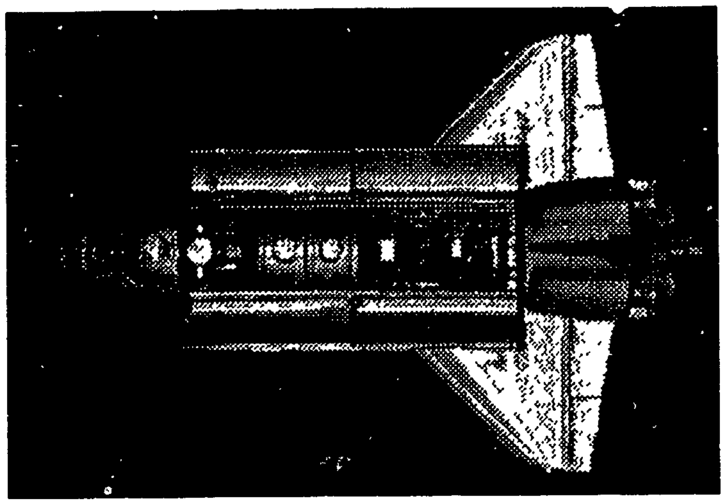
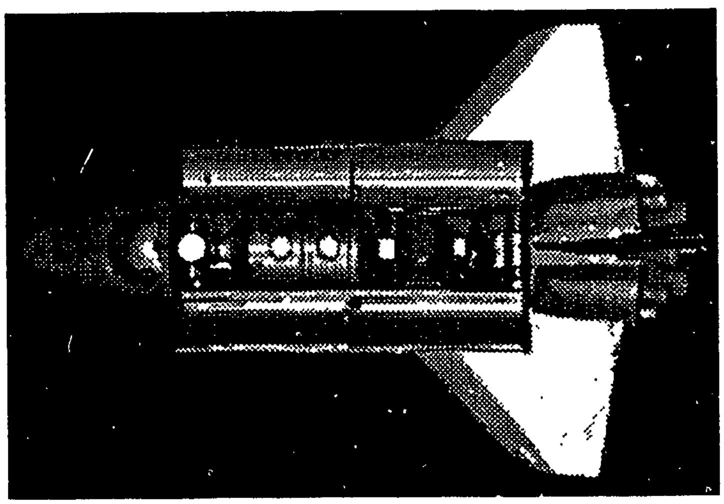


Figure 7.32: The space shuttle

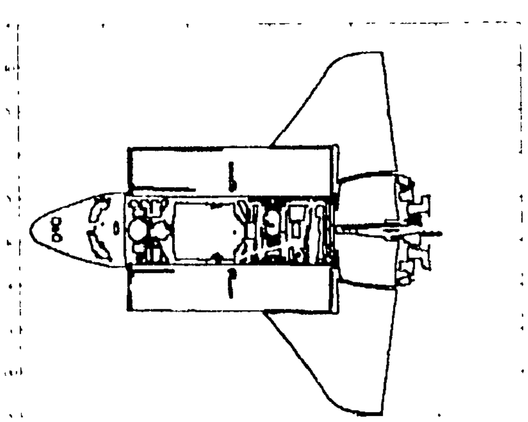
Original



Reconstruction



Preliminary segmentation



Final segmentation

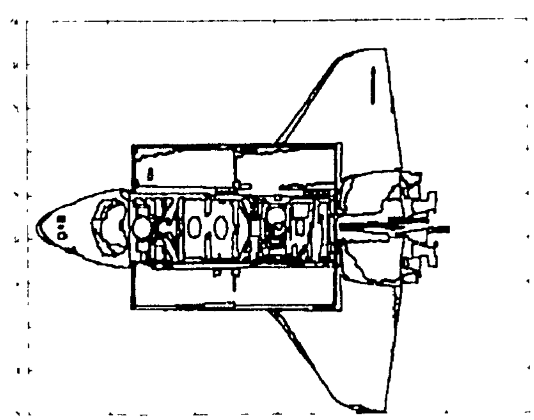
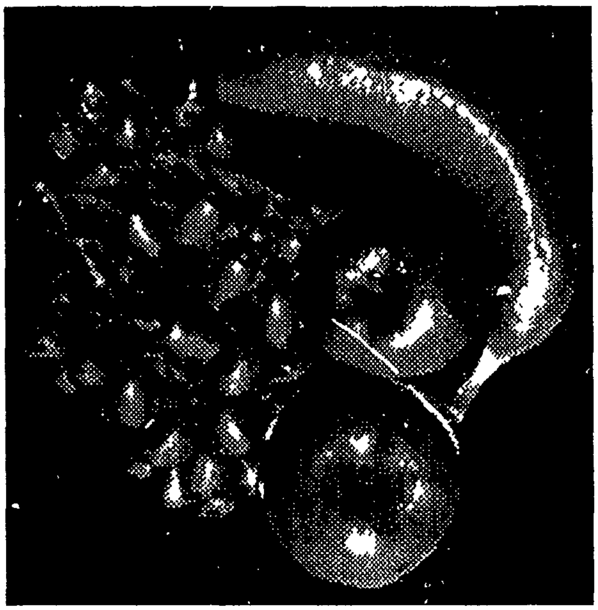
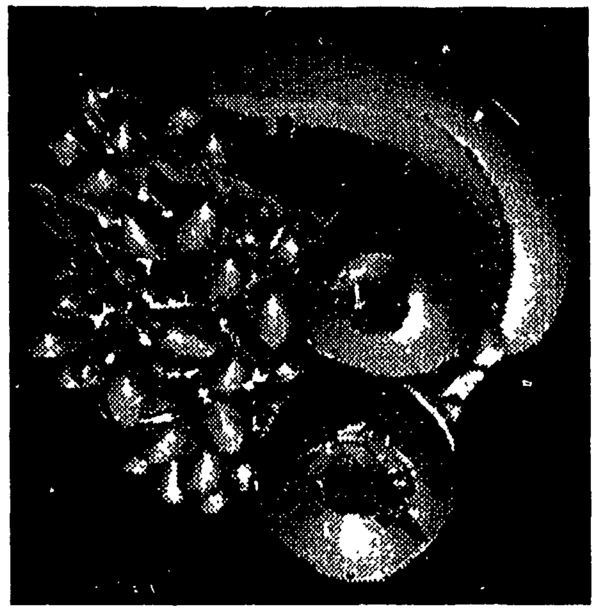


Figure 7.33: Fruit

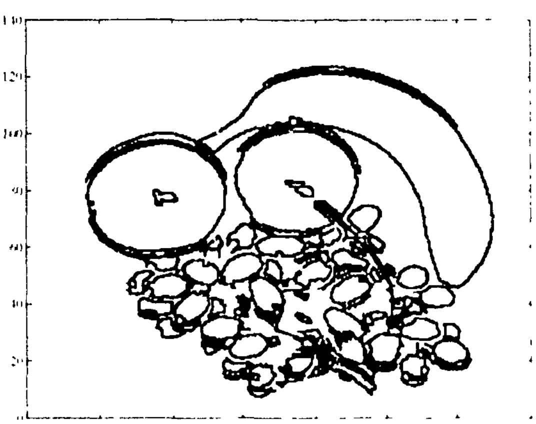
Original



Reconstruction



Preliminary Segmentation



Final Segmentation

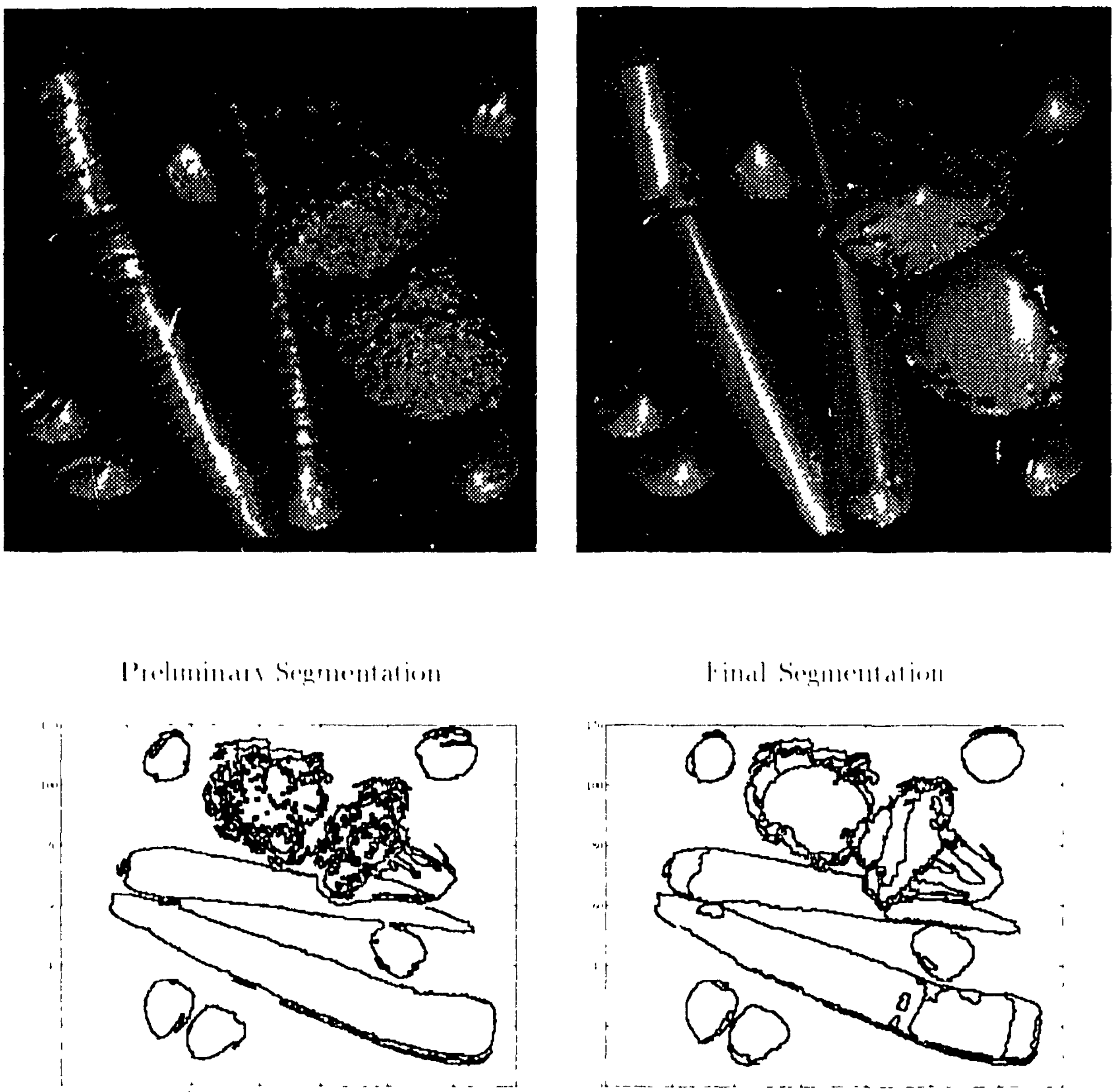
Figure 7.34: Vegetable

Original

Reconstruction

Preliminary Segmentation

Final Segmentation



for the first order primitive. The constant term J is set to 1 for the quadratic equation, therefore 9 points from range image are necessary to solve the equation. We list the parameters of some real range image experiments.

We used a realistic graphic rendering system developed in our laboratory to show range images. The object can be seen like a real one and it can be scaled, translated and rotated. In the experimental results, we show original range image, reconstructed range image, preliminary segmentation and final segmentation. For some simple object, we label the surface and list equation parameters of each surface patch. We now explain the meaning of each field in the list. The first column L is the label of the patch. It matches the label in the segmentation figure. N is the number of points in the patch. E is the average error in the patch. Equation parameters are values of $A, \ldots, I$ of Equation 7.6, when there are 9 values. They are listed by order in three lines. It is also indicate that the surface type is quadratic. When there are only three numbers, they are A, B and C in Equation 7.7 and indicate that the surface is a plane. The column V is for invariants of quadratic surface i.e. V_1, V_2 and V_3 in Equation C.8. The rotation matrix and translation vector (T) in the tables compose the pose of the patch. The detail explanations of the invariant and pose of quadratic surface can be found in Appendix C. Note that the range images from Michigan State University have different unit from that of NRC. Therefore we scale it by 20 in order to get similar range. For complex objects, although we have all parameters, it is very difficult to label it on the segmentation figure. On our computer, we can move cursor to each patch and click the mouse button to check all information on the screen.

Figure 7.27 shows the original 256 × 256 range image displayed in different perspective views with shaded surfaces. The noise effect can be clearly seen in these figures. Segmentation and fitting of the grip range image took 65 seconds of the CPU time using our implementation. The bottom figure in Figure 7.28 displays our segmentation of the range image. As we mentioned before, the edge and ...

regions are not fitted, resulting in gaps between fitted regions. The reconstruction in Figure 7.27 also shows the different views of the object. The reconstructed images are quite similar to the original except that they are noise-free and have gaps between the fitted surface patches. In Figure 7.32 the original range image of a space shuttle has resolution 512 × 400. Segmentation result and reconstruction is also displayed in Figure 7.32. The processing time for the space shuttle range image is about 400 seconds. Figure 7.30 shows a junk of objects. Figure 7.31 shows the tube. Nature objects are more interesting. Figure 7.33 shows some fruits and Figure 7.34 shows some vegetables. We see that second order surfaces have limitations to express these complex object. Also, the details of the object are considered as noise and fitted by a large smooth surface patch.

Besides the range images from NRC, we also tested several range images from MSU PRIP Lab in public domain. Figure 7.28 and Figure 7.29 display the segmentation and reconstruction of range images provided by the MSU PRIP Lab. Note that in Figure 7.29 there are no jump edges between the segments and no segment has more than 10% of the total area. In this case, RESC has succeeded in fitting each segment directly from the whole image. This demonstrates the robustness of the RESC method.

7.4 Summary

In this chapter, we have described the results of extensive synthetic experiments on the RESC algorithm and demonstrated the segmentation and fitting of the real range images. The experiments revealed that RESC method is highly robust to outliers. The highest breakdown point observed in the experiments is 91%. The breakdown point decreases with increasing noise levels. Curved surfaces are much more difficult to be estimated correctly. The breakdown point for curved surfaces is much lower than that for the planar surfaces. We found that when the input data were not contaminated by outliers, the LMS method was weak. The LMS method in this case ignores all upper part of residuals. The minimization is only based on the lower part of residuals. The RESC method has a better performance in such situations because the optimization is based on the whole inlier part.

In real range image experiments, we demonstrated the whole segmentation and fitting process. The reconstructed range images are almost exactly the same as the original except we cannot see any noise effect.

Comparison of segmentation results is difficult. There is no standard evaluation criterion. We measure our segmentation results by evaluate the fitting error of each surface patch. If the average errors are less than a given threshold, then the fitting is accepted. Our segmentation algorithm is based on the random sampling principle. The larger patches and the less noise patches are fitted first. Therefore our segmentation starts from the easiest patch.

Our segmentation is actually a primitive extraction process. It provides a very convenient tool for object recognition. When object model consists of only primitives, we can match primitives to the object model after segmentation.

Our robust fitting technique, unlike others [5, 7, 21], can tolerate a large percentage of outliers. In real range images outliers occur frequently. They occur especially when object surface has some tiny scratch or shiny patch. In these cases our system is robust.

Our range image rendering system produces impressive displays. It provides a way to verify the segmentation and fitting result by rendering the reconstructed range images. In the literature (e.g. [27, 5, 7, 21]) we found that most authors just display range images by mapping the depth of each pixel to its brightness. Some authors give segmentation results without reconstruction.

Our algorithm is very simple. It is very easy to implement. APL or [illegible] selecting some adjustable parameters, it works well with these parameter settings. In our experiments, all images were processed with the same parameter settings.

Comparing timing is difficult if not impossible.

Chapter 8

Conclusions and Directions for Future Research

We conclude the dissertation in section 8.1 and propose future research in section 8.2.

8.1 Conclusions

In this dissertation, we propose a new *high breakdown point robust estimation* method (RESC) and we apply it for primitive extraction from a data set. The RESC is a *substantial* improvement to the LMS and can be used in various areas where robust estimation is needed. Unlike LMS method, RESC allows raw data to have more than 80% of outliers, whereas LMS works only for the data with less than 50% of outliers. Histogram method makes residual statistics of each random sample and chooses the best among all samples which show *residual consensus*, maximum *histogram power*

The object function of the RESC process is histogram power. It considers two factors: the number of points on and near the primitive surface should be as large as possible and the residuals of the total inlier points should be as small as possible. By the combination of the two factors, RESC achieves better performance than LMS because evaluation at the median point only is a weak criteria of LMS. With this histogram method, the *inlier* part and the *outlier* part can be separated. *Compressed histogram* method works at different noise levels. The experimental results show that the RESC algorithm has much better performance than the least squares method and the least median squares method in the second order primitive estimation.

We apply the RESC method to range image segmentation and fitting. The RESC extracts first order and second order geometric primitives from range image. Each primitive is classified as a segment. The whole range image is then segmented into these primitive surface patches. It always segments the largest patch first because points from the largest patch are more likely to be chosen than those from smaller patches. Since standard deviation σ for each surface patch can be correctly estimated from the histogram, the segmentation is reliable even for smoothly connected curved regions.

The different primitive type switching is based on the validation of the sample points, the invariants extracted from quadratic surface parameters and the average curvatures. The method which assumes the second order primitive at the beginning and then switch it to the first order primitive if it belongs to the first order primitive can avoid repeated surface fitting for every region.

The experimental results for synthetic data and 3D range images are visually and quantitatively convincing.

A genetic algorithm (GA) is incorporated into our RESC method to accelerate the processing. Most genes in the literature are binary genes. Our genes are integers which are point indices of the input points in the current processing region. We analyzed the situation if such integer genes are expressed in binary form. Crossover operator will break, with very high probability, the integer, giving a new value for the integer. Such a break is equivalent to a mutation operation and the equivalent mutation rate is calculated for n-point crossover operator. We cannot control the mutation if the gene is expressed as binary digits, therefore, we do not use binary gene.

Although there is still no fundamental theory about the performance and convergence of GA, the empirical studies give a guideline for the selections of GA parameters. Two different GAs are tested. A steady state GA has much better performance than a generational replacement GA. The experimental results show that the mutation rate of the best settings is much higher than the range suggested by other researchers. Using integer gene in our GA is one reason for such high mutation rate. Upon a good selection of the mutation rate, population size is not a very sensitive factor. GA can work well over a large range of population size. The RESC algorithm works very well under the support of GA for the stochastic searching of the best sample points over the unsegmented range images.

The major contributions of the dissertation and the publications based on the dissertation are listed in Section 1.1 of Chapter 1.

8.2 Future Research

We propose the following research directions in the future:

- Robust estimation is very important not only in computer vision area, but also in statistics, mathematics, *etc.* Robustness of an estimator can never be over emphasized. The RESC method achieves one of the objectives: high breakdown point. In the future, a high efficiency should also be emphasized. The linear least squares method is highly efficient, but it is not robust to outliers. We have to pay a price for robustness by sacrificing efficiency.

- The RESC method is more complicated than the least squares method and the least median squares method. Several parameters must be determined properly in order for RESC to work well. It will be useful to simplify the algorithm and reduce the adjustable parameters.

- Genetic algorithms are incorporated in the RESC method and search speed is accelerated. It is necessary to investigate further on how the GA works and how to increase the efficiency of GA.

- The parameter settings of GA are determined by extensive experiments. We found that different situations may require different parameter settings and the best settings in our case is different from the suggested range given by other researchers in other situations. An easy way to determine the parameters should be explored to save time and effort.

- We use two orders of surface primitives to segment object surfaces. It works well for simple objects. It is difficult to express complex object using only two orders of primitives. A general purpose vision system should have more powerful means to represent surfaces. The other representations, such as splines, NURBS (Non Uniform Rational B-Spline), *etc.*, should be explored. A robust segmentation for these representations will be very useful.

- It is possible to extend the proposed method to grey level images. Range image has depth values for each pixels, therefore it is easier to process. Grey level images are more natural, however, since our eyes are more sensitive to color and brightness than depth. Range image is normally obtained by an active way using laser finders. This limits some of the applications where an active sensor is not possible.

- The recognition algorithm proposed in the dissertation is based on quadratic invariants. The decomposition of a complex surface into quadratic surfaces is not unique. This raises difficulty for the recognition based only on quadratic invariants. For a complete system, it has to consider other surface types other than planar and quadratic surfaces.

- The estimation of the invariants of a quadratic surface patch from noisy data is very difficult. All tested methods can tolerate only a small amount of noise. A little higher noise level may change the invariants drastically. A better estimator which is robust to noise is needed.

We believe that computer vision is one of the most challenging research areas in the next century. Many applications need further development of computer and sensor hardware. The fundamental theories of vision system should be more extensively explored. More efficient algorithms and more robust methods have to be developed. In the future we believe that computer vision will catch up and actually exceed the ability of human vision.

Bibliography

[1] John (Yiannis) Aloimonos. "Purposive and qualitative active vision". In *Proceedings of the 10th International Conference on Pattern Recognition*, volume 1, pages 346-360, Atlantic City, New Jersey, USA, June 16-21 1990.

[2] J. E. Baker. "Adaptive selection methods for genetic algorithms". In *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, pages 101-111, 1985.

[3] J. E. Baker. "Reducing bias and inefficiency in the selection algorithm". In *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 525-532, 1987.

[4] D. H. Ballard. "Generalizing the Hough Transform to Detect Arbitrary Shapes". *Pattern Recognition*, 13:111-122, 1981.

[5] Paul J. Besl. *Surfaces in Range Image Understanding*. Springer Verlag, 1988.

[6] Paul J. Besl and Ramesh C. Jain. "Three dimensional object recognition". *Computing Surveys*, 17(1):75-145, March 1985.

[7] Paul J. Besl and Ramesh C. Jain. "Segmentation through variable order surface fitting". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(2):167-192, March 1988.

[8] Robert C. Bolles and Martin A. Fischler. "A RANSAC based approach to model fitting and its application to finding cylinders in range data". In *Proceedings of*

the Seventh International Joint Conference on Artificial Intelligence IJCAI-81, pages 637–643, UBC, Vancouver, B.C. Canada, August 24–28 1981.

[9] Fred L. Bookstein. "Fitting conic sections to scattered data". *Computer Graphics and Image Processing*, 9:56–71, 1979.

[10] D. S. Chen. "a data-driven intermediate level feature extraction algorithm". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7), July 1989.

[11] Roland T. Chin and Charles R. Dyer. "Model based recognition in robot vision". *Computing Surveys*, 18(1):67–108, March 1986.

[12] L. Davis and S. Coombs. "Genetic algorithms and communication link speed design: theoretical considerations". In *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 252–256, 1987.

[13] L. Davis and F. Ritter. "Schedule optimization with probabilistic search". In *Proceedings of the 3rd IEEE Conference on Artificial Intelligence Application*, pages 231–236, 1987.

[14] Lawrence Davis. "Adapting operator probabilities in genetic algorithms". In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 61–69, George Mason University, Virginia, July 4–7 1989.

[15] Lawrence Davis. "Bit climbing, representational bias, and test suite design". In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 18–23, University of California at San Diego, 1991.

[16] Kenneth A. DeJong. *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, Department of Computer and Communication Sciences, University of Michigan, 1975.

[17] D. L. Donoho and P. J. Huber. "The notion of breakdown point". In P. Bickel, K. Doksum, and Jr. J. L. Hodges, editors, *A Festschrift for Erich Lehmann*. Wadsworth, Belmont, CA, 1983.

[18] R. O. Duda and P. E. Hart. "Use of the Hough transformation to detect lines and curves in pictures". *Communications of the ACM*, 15:11–15, 1972.

[19] R. O. Duda and P. E. Hart. "Use of the Hough transformation to detect lines and curves in pictures". *Communication of ACM*, 15(1):11–15, January 1972.

[20] A. C. Englander. "Machine learning of visual recognition using genetic algorithms". In *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, pages 197–201, 1985.

[21] Ting-Jun Fan, Gerard Medioni, and Ramakant Nevatia. "Segmented descriptions of 3-D surfaces". *IEEE Journal of Robotics and Automation*, RA-3(6):527–538, December 1987.

[22] O. D. Faugeras and M. Hebert. "The representation, recognition, and locating of 3-D objects". *The International Journal of Robotics Research*, 5(3):27–52, Fall 1986.

[23] O. D. Faugeras and M. Hebert. "The representation, recognition, and positioning of 3-D shapes from range data". In A. Rosenfeld, editor, *Techniques for 3-D Machine Perception*, pages 13–51. Elsevier Science Publishers B.V., 1986.

[24] M. A. Fischler and R. C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". *Communications of the ACM*, 24:381–395, 1981.

[25] J. M. Fitzpatrick, J. J. Grefenstette, and D. Van Gucht. "Image registration by genetic search". In *Proceedings of IEEE Southeast Conference*, pages 460–464, 1984.

[26] P. J. Flynn and A. K. Jain. "Surface classification: hypothesis testing and parameter estimation". In *Proceedings of IEEE 1988 Computer Vision and Pattern Recognition*, pages 261–267, Ann Arbor, Michigan, June 5-9 1988.

[27] Patrick Joseph Flynn. *CAD-Based computer vision: modeling and recognition strategies*. PhD thesis, Computer Science Department, Michigan State University, 1990.

[28] David Forsyth, Joseph L. Mundy, Andrew Zisserman, and Christopher M. Brown. "Projectively invariant representations using implicit algebraic curves". In O. Faugeras, editor, *Proceedings of the First European Conference on Computer Vision — ECCV 90*, pages 427–436, Antibes, France, April 1990. Springer-Verlag.

[29] David Forsyth, Joseph L. Mundy, Andrew Zisserman, Chris Coelho, Aaron Heller, and Charles Rothwell. "Invariant descriptors for 3-D object recognition and pose". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):971–991, October 1991.

[30] R. Gerardy. "Probabilistic finite state system identification". *International Journal of General Systems*, 8:229–242, 1982.

[31] A. M. Gillies. *Machine learning procedures for generating image domain feature detectors*. PhD thesis, University of Michigan, Ann Arbor, 1985.

[32] D. E. Glover. *Experimentation with an adaptive search strategy for solving a keyboard design configuration problem*. PhD thesis, University of Iowa, 1986.

[33] D. E. Goldberg. *Computer-aided gas pipeline operation using genetic algorithms and rule learning*. PhD thesis, University of Michigan, Ann Arbor, 1983.

[34] David E. Goldberg. *Genetic Algorithms in Searching, Optimization, and Machine Learning*. Addison Wesley, 1989.

[35] M. D. Gordon. *Adaptive subject indexing in document retrieval*. PhD thesis, University of Michigan, Ann Arbor, 1984.

[36] John J. Grefenstette. "Optimization of control parameters for genetic algorithms". *IEEE Transactions on System, Man and Cybernetics*, 16(1):122–128, January 1986.

[37] Ernest L. Hall, James B. K. Tio, Charles A. McPherson, and Firooz A. Sadjadi. "Measuring curved surfaces for robot vision". *IEEE Computer*, 15(12):42–54, 1982.

[38] Robert M. Haralick and Linda G. Shapiro. *Computer and Robot Vision*. Addison-wesley Publishing Company, 1992.

[39] Hill and Taylor. Model based image interpretation using genetic algorithm. In *Proceedings of British Machine Vision Conference*, pages 266–271, London, 1991. Springer Verlag.

[40] Richard Hoffman and Anil K. Jain. "Segmentation and classification of range images". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):608–620, September 1987.

[41] John H. Holland. *Adaptation in natural and artificial systems*. The University of Michigan Press, 1975.

[42] P. W. Holland and R. E. Welsch. "Robust regression using iteratively reweighted least squares". *Commun. Stat.*, A6:813–828, 1977.

[43] P. V. C. Hough. *A Method and Means for Recognizing Complex Patterns*. U.S. Patent No. 3,069,654, 1962.

[44] P. J. Huber. *Robust Statistics*. Wiley, New York, 1981.

[45] J. Illingworth and J. Kittler. "A survey of the Hough transform". *Computer Vision, Graphics and Image Processing*, 44(1):87–116, January 1988.

[46] Jr. J. L. Hodges. "Efficiency in normal samples and tolerance of extreme values for some estimates of location". In *Proceedings of Fifth Berkeley Symp. Math. Stat. Probab.*, volume 1, pages 163–168, 1967.

[47] Anil K. Jain and Sateesha G. Nadabar. "MRF model-based segmentation of range images". In *Proceedings of the Third International Conference on Computer Vision*, pages 667–671, Osaka, Japan, December 4-7 1990.

[48] Jean Michel Jolion, Peter Meer, and Samira Bataouche. "Robust clustering with applications in computer vision". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8):791–802, August 1991.

[49] Behzad Kamgar-Parsi, Behrooz Kamgar-Parsi, and Nathan S. Netanyahu. "A nonparametric method for fitting a straight line to a noisy image". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(9):998–1001, September 1989.

[50] T. Kasvand. "Extraction of Edges in 3D Range Images to Subpixel". In *Proceedings of the 9th International Conference on Pattern Recognition*, Rome, Italy, November 14-17 1988.

[51] T. Kasvand. "The K1K2 Space in Range Image Analysis". In *Proceedings of the 9th International Conference on Pattern Recognition*, Rome, Italy, November 14-17 1988.

[52] G. A. Korn and T. M. Korn. *Mathematical Handbook for Scientists and Engineers*. McGraw Hill, New York, N.Y., 1961.

[53] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Springer Verlag, 1992.

[54] Ping Liang. "A new transform for curve detection". In *Proceedings of the Third International Conference on Computer Vision*, pages 748–751, Osaka, Japan, December 4-7 1990.

[55] Peter Meer and Doron Mintz. "Robust regression in computer vision". In *SPIE Vol. 1381 Intelligent Robots and Computer Vision IX: Algorithms and Techniques (1990)*, pages 424–435, Boston, Massachusetts, USA, November 5–[illegible], 1990.

[56] Peter Meer, Doron Mintz, and Azriel Rosenfeld. "Robust regression methods for computer vision: a review". *International Journal of Computer Vision*, 6(1), April 1991.

[57] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1992.

[58] Doron Mintz, Peter Meer, and Azriel Rosenfeld. "Consensus by decomposition: a paradigm for fast high breakdown point robust estimation". Technical Report CAR-TR-525, Center for Automation Research, University of Maryland, College Park, MD 20742, USA, December 1990.

[59] Ben Noble and James W. Daniel. *Applied Linear Algebra*. Prentice-Hall, Inc., 1977.

[60] T. Pavlidis and S. L. Horowitz. "segmentation of plane curves". *IEEE Transactions on Computers*, C-23:860–870, 1974.

[61] Theo Pavlidis. *Algorithms for Graphics and Image Processing*. Computer Science Press, MD, U.S.A., 1982.

[62] Vaughan Pratt. "Direct least-squares fitting of algebraic surfaces". *ACM Computer Graphics*, 21(4):145–152, July 1987.

[63] V. V. Raghavan and B. Aggarwal. "Optimal determination of user oriented clusters: an application for the reproductive plan". In *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 241–246, 1987.

[64] L. A. Rendell. "Genetic plans and the probabilistic learning system: synthesis and results". In *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, pages 60–73, 1985.

[65] A. Rosenfeld. *Picture Processing by Computer*. Academic Press, New York, 1969.

[66] Gerhard Roth and Martin D. Levine. "Segmentation of geometric signals using robust fitting". In *Proceedings of the 10th International Conference on Pattern Recognition*, volume 1, pages 826–831, Atlantic City, New Jersey, USA, June 16-21 1990.

[67] Gerhard Roth and Martin D. Levine. "A genetic algorithm for primitive extraction". In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 487–494, University of California at San Diego, 1991.

[68] Gerhard Roth and Martin D. Levine. "Random sampling for primitive extraction". In *International Workshop on Robust Computer Vision*, Seattle, Washington, October 1991.

[69] Peter J. Rousseeuw and Annick M. Leroy. *Robust Regression and Outlier Detection*. John Wiley & Sons, Inc., New York, 1987.

[70] Bikash Sabata, Farshid Arman, and J. K. Aggarwal. "Segmentation of 3-D range images using pyramidal data structures". In *Proceedings of the Third International Conference on Computer Vision*, pages 662–666, Osaka, Japan, December 4-7 1990.

[71] Peter T. Sander and Steven W. Zucker. "Tracing surfaces for surfacing traces". In *Proceedings of the First International Conference on Computer Vision*, pages 241–249, London, England, June 8-11 1987.

[72] J. D. Schaffer. "Some effects of selection procedures on hyperplane sampling by genetic algorithms". In L. Davis, editor, *Genetic algorithms and simulated annealing*, pages 93–100. Pitman, London, 1987.

[73] J. David Schaffer, Richard A. Caruana, Larry J. Eshelman, and Rajarshi Das. "A study of control parameters affecting online performance of genetic algorithms for function optimization". In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 51-60, George Mason University, Virginia, July 4-7 1989.

[74] J. F. Silverman and D. B. Cooper. "Bayesian clustering for unsupervised estimation of surface and texture models". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10, July 1988.

[75] I. Stadnyk. "Schema recombination in pattern recognition problems". In *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 27-35, 1987.

[76] G. C. Stockman and A. K. Agrawala. "Equivalence of Hough curve detection to template matching". *Communications of the ACM*, 20:820-822, 1977.

[77] Gilbert Syswerda. "Uniform crossover in genetic algorithms". In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 2-9, George Mason University, Virginia, July 4-7 1989.

[78] Gabriel Taubin. "Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(11):1115-1138, November 1991.

[79] Darrell Whitley. "Using reproductive evaluation to improve genetic search and heuristic discovery". In John J. Grefenstette, editor, *Genetic Algorithms and Their Applications*. Lawrence Erlbaum Associates, 1987.

[80] Lei Xu, Erkki Oja, and Pekka Kultanen. "A new curve detection method: randomized Hough transform (RHT)". *Pattern Recognition Letters*, 11(5):331-338, May 1990.

[81] Naokazu Yokoya and Martin D. Levine. "Range image segmentation based on differential geometry: a hybrid approach". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6):643 , June 1989.

[82] Xinming Yu, T. D. Bui, and A. Krzyżak. "3D object recognition and pose determination by quadratic invariants". In *Proceedings of the 7th Scandinavian Conference on Image Analysis*, Aalborg University, Denmark, August 13-16 1991.

[83] Xinming Yu, T. D. Bui, and A. Krzyżak. "3D range image segmentation and fitting by quadratic surfaces". In *Proceedings of the SPIE Advances in Intelligent Robotic Systems, Sensor Fusion IV: Control Paradigms and Data Structures*, pages 576 585, Boston, USA, November 10 15 1991.

[84] Xinming Yu, T. D. Bui, and A. Krzyżak. "Robust contour decomposition by residual consensus". In *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, pages 1.5.1 1.5.4, Québec, Canada, September 25 27 1991.

[85] Xinming Yu, T. D. Bui, and A. Krzyżak. "Invariants and pose determination". In Carlo Arcelli, Luigi P. Cordella, and Gabriella Sanniti Di Baja, editors, *VISUAL FORM: Analysis and Recognition, Proceedings of the International Workshop on Visual Form, held May 27-30, 1991, in Capri, Italy*, pages 623 632, Plenum Press, New York, 1992.

[86] Xinming Yu, T. D. Bui, and A. Krzyżak. "Range image segmentation and fitting by residual consensus". In *Proceedings of IEEE 1992 Computer Vision and Pattern Recognition*, pages 657 660, Champaign, Illinois, June 15 18 1992.

[87] Xinming Yu, T. D. Bui, and A. Krzyżak. "Range Image Segmentation and Fitting using Robust Estimation with High Breakdown Point". Submitted to *The IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1992.

[88] Xinming Yu, T. D. Bui, and A. Krzyżak. "The Genetic Algorithm Parameter Settings for Robust Estimation and Range Image Segmentation and Fitting". In

Proceedings of the 8th Scandinavian Conference on Image Analysis, University of Tromsoe, Norway, May 25–28 1993.

[89] Xinhua Zhuang, Tao Wang, and Peng Zhang. "A highly robust estimator through partially likelihood function modeling and its application in computer vision". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(1):19–35, January 1992.

Appendix A

Derivation of Singular Matrix

When a second order primitive model is applied to a first order data set, then the matrix **X** in Equation (4.20) is singular. Here we derive two-dimensional cases. The results can be generalized to three dimensional situations. For convenience, we use symbol **A** instead of **X** in Equation (4.20). A two dimensional second order primitive model (from Table 3.1) is

$$a_1 x^2 + a_2 y^2 + a_3 xy + a_4 x + a_5 y = 1 \tag{A.1}$$

Matrix **A** in equation (4.20) in this case is

$$\mathbf{A} = \begin{bmatrix} x_1^2 & y_1^2 & x_1 y_1 & x_1 & y_1 \\ x_2^2 & y_2^2 & x_2 y_2 & x_2 & y_2 \\ x_3^2 & y_3^2 & x_3 y_3 & x_3 & y_3 \\ x_4^2 & y_4^2 & x_4 y_4 & x_4 & y_4 \\ x_5^2 & y_5^2 & x_5 y_5 & x_5 & y_5 \end{bmatrix} \tag{A.2}$$

Matrix **A** is singular if and only if $\det(\mathbf{A}) = 0$. We will calculate the determinant of

A under the condition that points $(x_1, y_1), \ldots, (x_5, y_5)$ lie on a line, i.e.

$$y - y_i = \frac{y_j - y_i}{x_j - x_i}(x - x_i), \qquad \text{for } i, j = 1, \ldots, 5, \text{ and } i \neq j \tag{A.3}$$

Here, $x_i \neq x_j$ for $i, j = 1, \ldots, 5$ and $i \neq j$ since if it is not, there must exist two coincident points, then **A** is singular by equation (4.22). The determinant of **A** is

$$\det(\mathbf{A}) = \begin{vmatrix} x_1^2 & y_1^2 & x_1y_1 & x_1 & y_1 \\ x_2^2 & y_2^2 & x_2y_2 & x_2 & y_2 \\ x_3^2 & y_3^2 & x_3y_3 & x_3 & y_3 \\ x_4^2 & y_4^2 & x_4y_4 & x_4 & y_4 \\ x_5^2 & y_5^2 & x_5y_5 & x_5 & y_5 \end{vmatrix} \tag{A.4}$$

Replace line i by (line i − line 1), for $i = 2, \ldots, 5$.

$$\det(\mathbf{A}) = \begin{vmatrix} x_1^2 & y_1^2 & x_1y_1 & x_1 & y_1 \\ x_2^2 - x_1^2 & y_2^2 - y_1^2 & x_2y_2 - x_1y_1 & x_2 - x_1 & y_2 - y_1 \\ x_3^2 - x_1^2 & y_3^2 - y_1^2 & x_3y_3 - x_1y_1 & x_3 - x_1 & y_3 - y_1 \\ x_4^2 - x_1^2 & y_4^2 - y_1^2 & x_4y_4 - x_1y_1 & x_4 - x_1 & y_4 - y_1 \\ x_5^2 - x_1^2 & y_5^2 - y_1^2 & x_5y_5 - x_1y_1 & x_5 - x_1 & y_5 - y_1 \end{vmatrix} \tag{A.5}$$

Extract factor $(x_i - x_1)$ from line i, for $i = 2, \ldots, 5$, and define

$$k = \frac{y_i - y_1}{x_i - x_1}, \quad \text{for } i = 2, \ldots, 5, \tag{A.6}$$

$$\alpha = (x_2 - x_1)(x_3 - x_1)(x_4 - x_1)(x_5 - x_1), \tag{A.7}$$

and notice that

$$\frac{x_i^2 - x_1^2}{x_i - x_1} = x_i + x_1 \quad \text{for } i = 2, \ldots, 5, \tag{A.8}$$

$$\frac{y_i^2 - y_1^2}{x_i - x_1} = \frac{y_i - y_1}{x_i - x_1}(y_i + y_1) = k(y_i + y_1) \quad \text{for } i = 2, \ldots, 5 \tag{A.9}$$

and

$$\frac{x_iy_i - x_1y_1}{x_i - x_1} = \frac{x_iy_i - x_1y_i + x_1y_i - x_1y_1}{x_i - x_1} = y_i + kx_1 \quad \text{for } i = 2, \ldots, 5 \tag{A.10}$$

The determinant becomes

$$\det(\mathbf{A}) = \alpha \begin{vmatrix} x_1^2 & y_1^2 & x_1y_1 & x_1 & y_1 \\ x_2 + x_1 & k(y_2+y_1) & y_2 + kx_1 & 1 & k \\ x_3 + x_1 & k(y_3+y_1) & y_3 + kx_1 & 1 & k \\ x_4 + x_1 & k(y_4+y_1) & y_4 + kx_1 & 1 & k \\ x_5 + x_1 & k(y_5+y_1) & y_5 + kx_1 & 1 & k \end{vmatrix} \qquad (A.11)$$

Replace column 5 by (column 5 − column 4 × k):

$$\begin{aligned} \det(\mathbf{A}) &= \alpha \begin{vmatrix} x_1^2 & y_1^2 & x_1y_1 & x_1 & y_1 - kx_1 \\ x_2 + x_1 & k(y_2+y_1) & y_2 + kx_1 & 1 & 0 \\ x_3 + x_1 & k(y_3+y_1) & y_3 + kx_1 & 1 & 0 \\ x_4 + x_1 & k(y_4+y_1) & y_4 + kx_1 & 1 & 0 \\ x_5 + x_1 & k(y_5+y_1) & y_5 + kx_1 & 1 & 0 \end{vmatrix} \\ &= \alpha(y_1 - kx_1) \begin{vmatrix} x_2 + x_1 & k(y_2+y_1) & y_2 + kx_1 & 1 \\ x_3 + x_1 & k(y_3+y_1) & y_3 + kx_1 & 1 \\ x_4 + x_1 & k(y_4+y_1) & y_4 + kx_1 & 1 \\ x_5 + x_1 & k(y_5+y_1) & y_5 + kx_1 & 1 \end{vmatrix} \end{aligned} \qquad (A.12)$$

By the property of determinant:

$$\begin{vmatrix} a_{11} & \cdots & a_{1i} + a'_{1i} & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2i} + a'_{2i} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & \cdots & a_{ni} + a'_{ni} & \cdots & a_{nn} \end{vmatrix} = \begin{vmatrix} a_{11} & \cdots & a_{1i} & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2i} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & \cdots & a_{ni} & \cdots & a_{nn} \end{vmatrix} + \begin{vmatrix} a_{11} & \cdots & a'_{1i} & \cdots & a_{1n} \\ a_{21} & \cdots & a'_{2i} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & \cdots & a'_{ni} & \cdots & a_{nn} \end{vmatrix}. \qquad (A.13)$$

the equation (A.12) can be expressed as:

$$\det(\mathbf{A}) = \alpha(y_1 - kx_1)(\mathbf{B} + \mathbf{C}). \qquad (A.14)$$

where

$$\mathbf{B} = \begin{vmatrix} x_2 + x_1 & k(y_2 + y_1) & y_2 & 1 \\ x_3 + x_1 & k(y_3 + y_1) & y_3 & 1 \\ x_4 + x_1 & k(y_4 + y_1) & y_4 & 1 \\ x_5 + x_1 & k(y_5 + y_1) & y_5 & 1 \end{vmatrix} \quad \text{(A.15)}$$

and

$$\mathbf{C} = \begin{vmatrix} x_2 + x_1 & k(y_2 + y_1) & kx_1 & 1 \\ x_3 + x_1 & k(y_3 + y_1) & kx_1 & 1 \\ x_4 + x_1 & k(y_4 + y_1) & kx_1 & 1 \\ x_5 + x_1 & k(y_5 + y_1) & kx_1 & 1 \end{vmatrix} \quad \text{(A.16)}$$

C = 0 because the third column and the fourth column in **C** are proportional. Split **B** again:

$$\mathbf{B} = (\mathbf{D} + \mathbf{E}) \quad \text{(A.17)}$$

where

$$\mathbf{D} = \begin{vmatrix} x_2 + x_1 & ky_2 & y_2 & 1 \\ x_3 + x_1 & ky_3 & y_3 & 1 \\ x_4 + x_1 & ky_4 & y_4 & 1 \\ x_5 + x_1 & ky_5 & y_5 & 1 \end{vmatrix} \quad \text{(A.18)}$$

and

$$\mathbf{E} = \begin{vmatrix} x_2 + x_1 & ky_1 & y_2 & 1 \\ x_3 + x_1 & ky_1 & y_3 & 1 \\ x_4 + x_1 & ky_1 & y_4 & 1 \\ x_5 + x_1 & ky_1 & y_5 & 1 \end{vmatrix} \quad \text{(A.19)}$$

D = 0 because column 2 and column 3 in **D** are proportional, and **E** = 0 because column 2 and column 4 in **E** are proportional. Therefore, we have proved

$$\det(\mathbf{A}) = 0. \quad \text{(A.20)}$$

Appendix B

Curvature Calculation of Quadratic Surfaces

The general Quadratic surface equation can be represented by

$$F(x,y,z) = q_1x^2 + q_2y^2 + q_3z^2 + q_4xy + q_5xz + q_6yz + q_7x + q_8y + q_9z + q_{10} = 0 \quad (B.1)$$

Differentiate equation (B.1).

$$\begin{aligned} F_x &= 2q_1x + q_4y + q_5z + q_7 \\ F_y &= 2q_2y + q_4x + q_6z + q_8 \\ F_z &= 2q_3z + q_5x + q_6y + q_9. \end{aligned} \quad (B.2)$$

For range image, z normally is expressed as a function of x and y:

$$z = f(x,y). \quad (B.3)$$

Therefore, if $F_z \neq 0$[1],

$$f_x = -F_x/F_z \qquad f_y = -F_y/F_z \tag{B.4}$$

The second order of differentiation of $f(x,y)$ is:

$$f_{xx} = -\frac{F_z(F_{xx} + F_{xz}f_x) - F_x(F_{xz} + F_{zz}f_x)}{F_z^2} \tag{B.5}$$

$$f_{yy} = -\frac{F_z(F_{yy} + F_{yz}f_y) - F_y(F_{yz} + F_{zz}f_y)}{F_z^2} \tag{B.6}$$

$$f_{xy} = -\frac{F_z(F_{xy} + F_{xz}f_y) - F_x(F_{yz} + F_{zz}f_y)}{F_z^2} \tag{B.7}$$

The Gaussian curvature K and the mean curvature H can be calculated as

$$K = \frac{LN - M^2}{EG - F^2} \tag{B.8}$$

$$H = \frac{EN + GL - 2FM}{2(EG - F^2)} \tag{B.9}$$

where

$$E = 1 + f_x^2, \tag{B.10}$$

$$F = f_x f_y, \tag{B.11}$$

$$G = 1 + f_y^2, \tag{B.12}$$

$$L = \frac{f_{xx}}{\sqrt{1 + f_x^2 + f_y^2}}, \tag{B.13}$$

$$M = \frac{f_{xy}}{\sqrt{1 + f_x^2 + f_y^2}}, \tag{B.14}$$

$$N = \frac{f_{yy}}{\sqrt{1 + f_x^2 + f_y^2}} \tag{B.15}$$

[1] For range image, this condition is always true because $F_z = 0$ only for surface normal parallel to the x-y plane and this is impossible in range image.

Appendix C

Invariants and Pose Determination of Quadratic Surfaces

In this appendix, we derive the *invariants* and *pose* of quadratic surfaces. The invariants of quadratic surface can be extracted to represent the shape. The pose matrix can then be determined.

C.1 Invariants and Pose of Quadratic Surface

C.1.1 Diagonalization by Rotation Matrix

Quadratic surface can be represented by

$$q_{11}x^2 + q_{22}y^2 + q_{33}z^2 + 2q_{12}xy + 2q_{23}yz + 2q_{13}xz + 2q_{14}x + 2q_{24}y + 2q_{34}z + q_{44} = 0 \quad (C.1)$$

or in matrix form :

$$\mathbf{X}'\mathbf{Q}\mathbf{X} = 0$$

where $\mathbf{X} = [x\ y\ z\ 1]'$ is a vector and

$$\mathbf{Q} = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ q_{41} & q_{42} & q_{43} & q_{44} \end{bmatrix}$$

is a *symmetric* matrix.

We will transform the quadratic equation to form *shape* parameters which are invariant under transformations and *pose* parameters which can be used for further processing. By linear algebra, matrix $\mathbf{Q}$ of the quadratic equation can be transformed to a diagonal matrix. We express $\mathbf{Q}$ in the block matrix form

$$\mathbf{Q} = \begin{bmatrix} \mathbf{q} & \mathbf{u} \\ \mathbf{u}' & q_{44} \end{bmatrix}$$

where $\mathbf{q}$ is a 3×3 symmetric matrix and $\mathbf{u}$ is a 3×1 column vector. Let $\lambda_1, \lambda_2, \lambda_3$ denote the eigenvalues of $\mathbf{q}$, and let $\mathbf{\Lambda}$ be a diagonal matrix with λ's as its elements. By a *unitary transformation* [59], there exists a 3×3 orthonormal matrix $\mathbf{r}$ such that

$$\mathbf{\Lambda} = \mathbf{r}'\mathbf{q}\mathbf{r} \quad (C.2)$$

where columns of $\mathbf{r}$ are normalized eigenvectors. We use *Jacobian* method to calculate the eigenvalues of $\mathbf{q}$ and the orthonormal matrix $\mathbf{r}$. The eigenvalues of $\mathbf{q}$ are unique for a given quadratic equation if we disregard the order. But for a quadratic surface matrix, $\mathbf{Q}$ is not unique. If we have equation $\mathbf{X}^t\mathbf{Q}\mathbf{X} = 0$, then the equation $\mathbf{X}^t(\alpha\mathbf{Q})\mathbf{X} = 0$ also holds, for any *scale factor* $\alpha \neq 0$. To eliminate the scale factor, we choose λ from the λ_i's to be of the largest magnitude and set : $\lambda'_i = \lambda_i/\lambda$, for $i = 1, 2, 3$. Assume that quadratic equation is not degenerated to a planar equation, therefore, $\lambda \neq 0$. We sort λ'_is in *descending* order, and arrange eigenvectors (columns in $\mathbf{r}$) *accordingly*. After such arrangement, λ'_is, are *invariant* under rotation and translation and $\lambda'_1 = 1$

Geometric significance of the process is that the surface is transformed to a new coordinate system whose axes are along the *principal axes* of the quadratic surface. The eigenvectors correspond to the principal axes. Because principal axes are non-directional, the coordinate axis can take any of the *two* directions along the principal axes

The following theorem states that the rotation matrix $\mathbf{r}$ in equation C.2 is not unique.

Theorem 1 *There are four rotation matrices which can diagonalize symmetric quadratic matrix* $\mathbf{q}$ *if* $\mathbf{q}$ *has distinct eigenvalues.*

Proof : Suppose that $\mathbf{q}$ has 3 distinct eigenvectors : $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$. Matrix $\mathbf{r}$ can be a combination of them : $\mathbf{r} = [\pm\mathbf{x}_1| \pm \mathbf{x}_2| \pm \mathbf{x}_3]$. Among these 8 different $\mathbf{r}$'s, four of them are rotation matrices ($det(\mathbf{r}) = 1$) and the others are reflection matrices ($det(\mathbf{r}) = -1$). Only rotation matrices are possible solutions in a real situation. If eigenvalues are not distinct, i.e. the multiplicity of the eigenvalues is greater than one, then there will be *unlimited number* of orthonormal matrices satisfying equation (C.2). We call this a *degenerate* case and analyze it in Appendix D.

When we calculate eigenvectors with *Jacobian* method, $\mathbf{r}$ is a real rotation matrix. Only after we rearrange eigenvectors in sorting eigenvalues, $\mathbf{r}$ is possibly changed to a reflection matrix. Therefore, in practice we count the number of column exchanges of **r** in the sorting procedure, if the number is odd then **r** has been changed to a reflection matrix and we have to multiply **r** by −1 to change it back, otherwise it is still a rotation matrix. We can express the four different **r**'s as

$$\mathbf{r}^{(i)} = \mathbf{r}\mathbf{g}_i, \qquad \text{for } i = 0, \ldots, 3 \tag{C.3}$$

where $\mathbf{g}_i$ is a rotation matrix defined by a modified unit matrix with its columns other than i's multiplied by −1.

C.1.2 Translation Matrix

In the diagonalization process, **q** is rotated by matrix **r** to a diagonal matrix **Λ**. We need to simplify it further to eliminate the first order terms r, q and [illegible]. Extend **r** to a 4 × 4 matrix:

$$\mathbf{R} = \begin{bmatrix} \mathbf{r} & \mathbf{o} \\ \mathbf{o}' & 1 \end{bmatrix} \tag{C.4}$$

where **o** is a *column zero vector* (3 × 1). If **Λ** has a *full* rank, i.e. **Q** is *nondegenerate* (various degenerate cases are analyzed in Appendix D), then $\mathbf{\Lambda}^{-1}$ exist. By defining the translation matrix **V**:

$$\mathbf{V} = \begin{bmatrix} \mathbf{e} & -\mathbf{\Lambda}^{-1}\mathbf{r}'\mathbf{u} \\ \mathbf{o}' & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{e} & \mathbf{v} \\ \mathbf{o}' & 1 \end{bmatrix}$$

where **e** denotes a 3 × 3 unit matrix and $\mathbf{v} = -\mathbf{\Lambda}^{-1}\mathbf{r}'\mathbf{u}$ is a *translation vector* (3 × 1), we can get the *standard quadratic form* **F**

$$\mathbf{F} = \mathbf{V}'\mathbf{R}'\mathbf{Q}\mathbf{R}\mathbf{V} = \begin{bmatrix} \mathbf{\Lambda} & \mathbf{o} \\ \mathbf{o}' & \lambda_t \end{bmatrix} \tag{C.5}$$

where

$$\lambda_t = q_{44} - \mathbf{u}'\mathbf{r}\mathbf{\Lambda}^{-1}\mathbf{r}'\mathbf{u} \tag{C.6}$$

and **F** is a diagonal matrix. Translation vector **v** is independent of the scale factor because for $\alpha\mathbf{Q}$, $\mathbf{v} = -(\mathbf{\Lambda}^{-1}/\alpha)\mathbf{r}'(\alpha\mathbf{u}) = -\mathbf{\Lambda}^{-1}\mathbf{r}'\mathbf{u}$. Also if d is calculated from **Q** is equation (C.6), then $\alpha\lambda_d$ results from $\alpha\mathbf{Q}$. To eliminate scale factor α and to make λ_d unique, we set

$$\lambda'_d = \lambda_d/\lambda. \qquad \text{(C.7)}$$

Note that even if we replace $\mathbf{r}$ with $\mathbf{r}^{(i)}$'s, $\mathbf{r}^{(i)} = \mathbf{r}\mathbf{g}_i$, for $i = 0, \ldots, 3$, as expressed in equation (C.3), the multiplication of $\mathbf{r}^{(i)}\mathbf{\Lambda}^{-1}(\mathbf{r}^{(i)})^t$ has the same value for different i's. This is easy to verify.

C.1.3 Invariants and Pose Matrix

Now, we can conclude the following :

Theorem 2 *Four invariants of a quadratic surface under translation and rotation are:*

$$\mathbf{I} = [\lambda'_1 \ \lambda'_2 \ \lambda'_3 \ \lambda'_d] \qquad \text{(C.8)}$$

They are independent of the scale factor of quadratic equations.

Because λ'_1 is normalized to 1, we need only 3 parameters to determine a given quadratic surface. To make the expression simple, we define a *normalized* standard quadratic matrix (*invariant matrix*) as $\mathbf{S} = \mathbf{F}/\lambda$, where $\mathbf{S}$ is a 4×4 diagonal matrix and the elements of $\mathbf{I}$ are the diagonal elements of $\mathbf{S}$. Let

$$\mathbf{P} = \mathbf{R}\mathbf{V} = \begin{bmatrix} \mathbf{r} & \mathbf{r}\mathbf{v} \\ \mathbf{o}^t & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{r} & \mathbf{w} \\ \mathbf{o}^t & 1 \end{bmatrix}.$$

where $\mathbf{w} = \mathbf{r}\mathbf{v} = -\mathbf{r}\mathbf{\Lambda}^{-1}\mathbf{r}^t\mathbf{u}$ remains constant when we replace $\mathbf{r}$ with $\mathbf{r}^{(i)}$'s. We call $\mathbf{P}$ the *pose matrix*, and express equation (C.5) as $\mathbf{F} = \mathbf{P}^t\mathbf{Q}\mathbf{P}$ and normalized standard matrix $\mathbf{S}$ as $\mathbf{S} = (\mathbf{P}^t\mathbf{Q}\mathbf{P})/\lambda = \mathbf{P}^t\bar{\mathbf{Q}}\mathbf{P}$, where $\bar{\mathbf{Q}} = \mathbf{Q}/\lambda$. Now we have split the original $\mathbf{Q}$ into *shape* parameter $\mathbf{S}$ and *pose matrix* $\mathbf{P}$. Since there are four different $\mathbf{r}$'s in equation (C.3), there are also four pose matrices $\mathbf{P}$'s accordingly. The four $\mathbf{P}$'s can be expressed as :

$$\mathbf{P}^{(i)} = \begin{bmatrix} \mathbf{r}\mathbf{g}_i & \mathbf{w} \\ \mathbf{o}^t & 1 \end{bmatrix}, \qquad \text{for } i = 0, \ldots, 3 \qquad \text{(C.9)}$$

Appendix D

Quadratic Surface Classification

Table D.1: Surfaces of the second order with no point of symmetry

Normal form : $\lambda_1 x^2 + \lambda_2 y^2 + m z + n = 0$

no.	λ_1	λ_2	m	name of surface
1	>0	>0	<0	elliptic paraboloid
2	>0	<0	<0	hyperbolic paraboloid (saddle surface)
3	>0	$=0$	$\neq 0$	parabolic cylinder

Table D.2: Surfaces of the second order with a point of symmetry

Normal form : $\lambda_1 x^2 + \lambda_2 y^2 + \lambda_3 z^2 + d = 0$

no.	λ_1	λ_2	λ_3	d	name of surface
1	>0	>0	>0	<0	ellipsoid
2	>0	>0	>0	>0	imaginary quadric
3	>0	>0	>0	=0	degenerate ellipsoid (single point)
4	>0	>0	<0	<0	hyperboloid of one sheet
5	>0	>0	<0	>0	hyperboloid of two sheets
6	>0	>0	<0	=0	elliptic double cone
7	>0	>0	=0	>0	cylinder with imaginary generators
8	>0	>0	=0	<0	elliptic cylinder
9	>0	>0	=0	=0	pair of intersecting imaginary planes
10	>0	<0	=0	≠0	hyperbolic cylinder
11	>0	<0	=0	=0	pair of intersecting real planes
12	>0	=0	=0	<0	2 parallel planes
13	>0	=0	=0	>0	2 imaginary parallel planes
14	>0	=0	=0	=0	coordinate plane (yz - plane)

Appendix E

Degenerate Cases

Quadratic parameters may be in *degenerate* form in which the *rank* of **A** is less than 3, i.e. there is at least one eigenvalue equal to 0, or in the case of *multiplicity* of eigenvalues greater than *one*. If rank of **A** is 1 or 0, quadratic surface may even degenerate into planar surface. Here we assume that the quadratic surfaces are not degenerated to the planar surfaces.

E.1 Rank of Λ is Less Than Three

Let $\mathbf{A} = \mathbf{R}^t\mathbf{Q}\mathbf{R}$, where $\mathbf{R}$ is from equation (C.1). When $rank(\Lambda) < 3$, $\mathbf{A}$ is of the form :

$$\mathbf{A} = \begin{bmatrix} \lambda_1 & 0 & 0 & a_{14} \\ 0 & \lambda_2 & 0 & a_{24} \\ 0 & 0 & 0 & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}.$$

Table E.1: Translation vector and invariants in degenerate cases

Rank(Λ)	Type	**v**	**I**
2	$a_{34} = 0$ elliptic cylinder hyperbolic cylinder	$-a_{14}/\lambda_1$ $-a_{24}/\lambda_2$ n/a	λ_1', λ_2'
2	$a_{34} \neq 0$ elliptic paraboloid hyperbolic paraboloid	$-a_{14}/\lambda_1$ $-a_{24}/\lambda_2$ $-(a_{44} - a_{14}^2/\lambda_1 - a_{24}^2/\lambda_2)/(2a_{34})$	$\lambda_1', \lambda_2', a_{34}/\lambda$
1	parabolic cylinder	$-a_{14}/\lambda_1$ n/a $-(a_{44} - a_{14}^2/\lambda_1)/(2a_{34})$	$\lambda_1', a_{34}/\lambda$

Here we assume $\lambda_3 = 0$ when $rank(\Lambda) = 2$, and $\lambda_2 = \lambda_3 = 0$ but $a_{34} \neq 0$ when $rank(\Lambda) = 1$. In Table E.1, we list surface type, translation vector **v** and invariants **I** in various degenerate cases. 'n/a' in translation vector means that the translation in this direction is not applicable because of the surface property.

E.2 Multiplicity of Eigenvalues is Greater Than One

Although in this case eigenvalues may not equal to zero, the rotation matrix cannot be determined because of symmetry of the surface. Examples of this case are *circular cylinder*, *elliptic cylinder*, etc. The eigenvector corresponding to the distinct eigenvalue is along direction of the symmetry axis, the other two eigenvectors are orthogonal to the symmetry axis. Unfortunately, there is unlimited number of such eigenvectors satisfying this condition. We cannot determine the whole transformation matrix in this case from just one surface patch.

In the case of *sphere*, the multiplicity of the eigenvalues is three. It is obvious that we cannot determine orientation of the sphere, but we can determine the position of the center of the sphere. In most cases, there are patches other than sphere in the scene. By combinations with more than one such case, we can still determine object pose.