Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

ROBUST APPROACH TO OBJECT RECOGNITION THROUGH FUZZY CLUSTERING AND HOUGH TRANSFORM BASED METHODS

by
Tianxiong Fu

Object detection from two dimensional intensity images as well as three dimensional range images is considered. The emphasis is on the robust detection of shapes such as cylinders, spheres, cones, and planar surfaces, typically found in mechanical and manufacturing engineering applications. Based on the analyses of different HT methods, a novel method, called the Fast Randomized Hough Transform (FRHT) is proposed. The key idea of FRHT is to divide the original image into multiple regions and apply random sampling method to map data points in the image space into the parameter space or feature space, then obtain the parameters of true clusters. This results in the following characteristics, which are highly desirable in any method: high computation speed, low memory requirement, high result resolution and infinite parameter space. This project also considers use of fuzzy clustering techniques, such as Fuzzy C Quadric Shells (FCQS) clustering algorithm but combines the concept of "noise prototype" to form the Noise FCQS clustering algorithm that is robust against noise. Then a novel integrated clustering algorithm combining the advantages of FRHT and NFCQS methods is proposed. It is shown to be a robust clustering algorithm having the distinct advantages such as: the number of clusters need not be known in advance, the results are initialization

independent, the detection accuracy is greatly improved, and the computation speed is very fast. Recent concepts from robust statistics, such as least trimmed squares estimation (LTS), minimum volume ellipsoid estimator (MVE) and the generalized MVE are also utilized to form a new robust algorithm called the generalized LTS for Quadric Surfaces (GLTS-QS) algorithm is developed. The experimental results indicate that the clustering method combining the FRHT and the GLTS-QS can improve clustering performance. Moreover, a new cluster validity method for circular clusters is proposed by considering the distribution of the points on the circular edge. Different methods for the computation of distance of a point from a cluster boundary, a common issue in all the range image clustering algorithms, are also discussed. The performance of all these algorithms is tested using various real and synthetic range and intensity images. The application of the robust clustering methods to the experimental granular flow research is also included.

ROBUST APPROACH TO OBJECT RECOGNITION THROUGH FUZZY CLUSTERING AND HOUGH TRANSFORM BASED METHODS

by
Tianxiong Fu

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

Department of Mechanical and Industrial Engineering

January 1995

APPROVAL PAGE

ROBUST APPROACH TO OBJECT RECOGNITION THROUGH FUZZY CLUSTERING AND HOUGH TRANSFORM BASED METHODS

Tianxiong Fu

Dr. Rajesh N. Dave, Dissertation Advisor — Date
Associate Professor of Mechanical Engineering, NJIT

Dr. Rong Chen, Committee Member — Date
Professor of Mechanical Engineering
and Associate Chairperson for Graduate Studies, NJIT

Dr. Sunil Dhar, Committee Member — Date
Associate Professor of Mathematics, NJIT

Dr. Zhiming Ji, Committee Member — Date
Assistant Professor of Mechanical Engineering, NJIT

Dr. Nouri Levy, Committee Member — Date
Associate Professor of Mechanical Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Tianxiong Fu

Degree: Doctor of Philosophy in Mechanical Engineering

Date: January 1995

Undergraduate and Graduate Education:

- Doctor of Philosophy in Mechanical Engineering,
  New Jersey Institute of Technology,
  Newark, New Jersey, 1995

- Master of Science in Mechanical Engineering,
  Huazhong University of Science and Technology,
  Wuhan, P. R. China, 1987

- Bachelor of Science in Mechanical Engineering,
  Huazhong University of Science and Technology,
  Wuhan, P. R. China, 1984

Major: Mechanical Engineering

Presentations and Publications:

- Rajesh N. Dave and Tianxiong Fu, "Robust shape detection using fuzzy clustering: practical applications," *Fuzzy Sets and Systems*, **65**, pp. 161-185, 1994.

- Tianxiong Fu and Rajesh N. Dave, "Segmenting curved surfaces from range images," *Advances in Manufacturing Systems: Design, Modeling and Analysis,* Elsevier, pp. 347-352, 1994.

This dissertation is dedicated to
my parents, my wife and my daughter.

ACKNOWLEDGMENT

I wish to express my sincere gratitude to my advisor, Dr. Rajesh N. Dave, for his expertise, guidance, and support throughout this research.

Special thanks to Drs. Rong Chen, Sunil Dhar, Zhiming Ji and Nouri Levy for serving as committee members.

I acknowledge the contributions of my colleagues Kurra Bhaswan and Jian Yu. I also acknowledge Dr. Raghu Krishnapuram of University of Missouri-Columbia and his research group for providing some image data and the code for data generation, and thank the PRIP Lab of Michigan State University for providing some image data.

I am grateful to the Particulate Flow Research Lab for providing computing facilities and convenience during the course of this research.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

LIST OF TABLES

LIST OF TABLES
(Continued)

Table **Page**

LIST OF FIGURES

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Purpose of Research

In the discrete product manufacturing industries, the most automated form of production is the computer-integrated manufacturing system (CIMS). Computer-integrated manufacturing systems are designed to fill the gap between high-production transfer lines and low-production NC machines. High-production transfer lines are used to produce the parts in large volumes due to their high efficiency. One transfer line can only manufacture one part at a time, which makes it very inflexible, namely, the highly mechanized line has to be shut down and redesigned when a minor change of part design is required, and sometimes, even the transfer line has to be discarded when the change is a major one. On the other hand, stand-alone NC machines possess very high flexibility, which are ideally suited for variations in workpart configuration. Numerically controlled machine tools are widely used in job shops and small batch manufacturing environments because they can be easily reprogrammed to deal with product changeovers and part design changes. As a trade off, NC machines can not produce machine parts at high output rates. In terms of manufacturing efficiency and productivity, there exists a gap between the high-production-rate transfer machines and the highly flexible NC machines. Nowadays, customers requirements to products are of a rich variety, so the manufacturers have to continually change and improve their designs to keep their share of the markets. Usually these parts

are of fairly complex geometry that belong to midrange production volumes, and the production equipment must be flexible enough to handle a variety of part designs. Transfer lines are not suited to this application because they are inflexible. NC machines are not suited to this application because their production rates are too slow. The solution to this midvolume production problem is the CIMS. CIMS can increase machine utilization, reduce direct and indirect labor, reduce manufacturing lead time, lower in-process inventory, and increase schedule flexibility [Groover et al. 1984].

Robots and mobile vehicles play an important role in the computer-integrated manufacturing systems. They are used to handle materials, such as transferring parts between machines, changing tools, loading and unloading parts, etc.. They make the systems more flexible. Unfortunately, because mobile vehicles and robots lack more intelligent guidance systems, their application ranges are greatly restricted. Computer vision is an ideal tool to enhance intelligence to those systems.

In modern manufacturing technology, computer-aided quality control is an important aspect. It is concerned with those activities related to inspection of product and component quality, detection of poor quality, and corrective action necessary to eliminate poor quality. Among many different inspection methods, computer vision is an attractive noncontact method. The use of computer vision system for inspection is an exciting area which holds the promise of significant improvements in both the productivity of the inspection process and the quality of the resulting product. Computer vision systems can deal with noncontact gaging of dimensions as well as inspection based on pattern recognition of object features.

Moreover, computer vision systems can be used for scientific and medical image analysis, classifying biological cells is an application example. They can also be used in aerospace and defense industries, such as target tracking, simulation and reconnaissance.

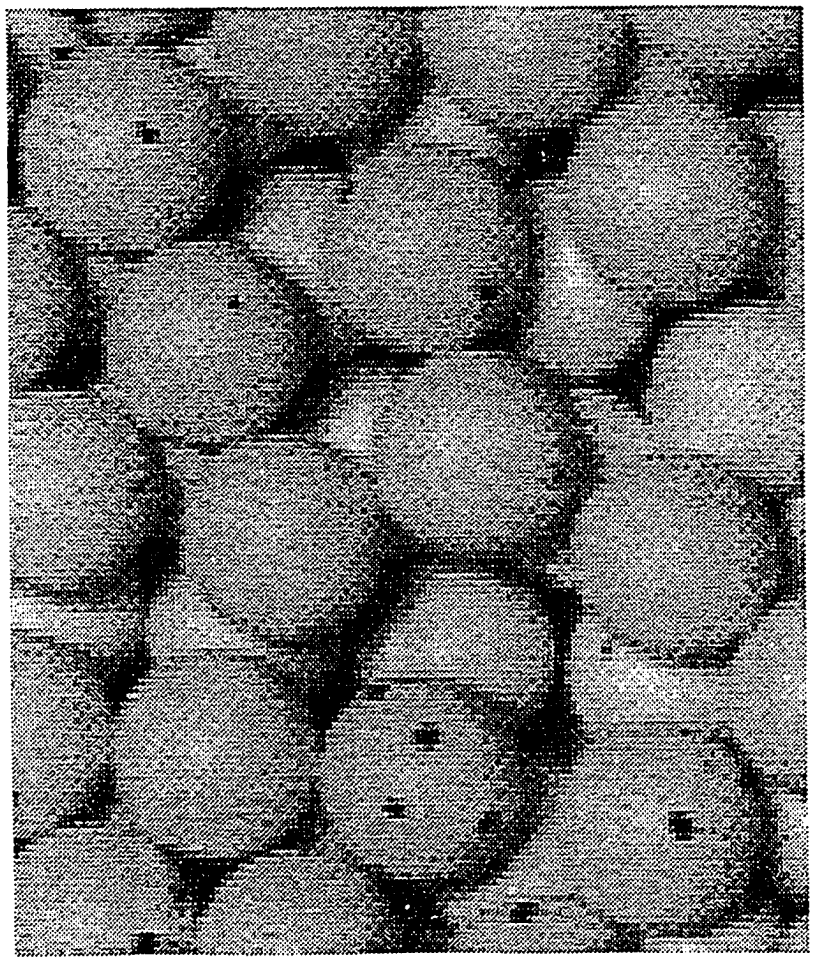


Figure 1.1 Image of Randomly Packed Spheres.

There are other areas in engineering where computer vision plays an important role. For example, recently, there has been a considerable increase in the use of vision based techniques for experimental studies in multi-phase flows [Ghidaglia et al.1993, Hsiau and Hunt 1993, Krishnan et al. 1992, Li et al. 1992]. These applications include measurement of velocity fields, measuring flow patterns, describing the flow structure (including micro-structure of particles), and finding velocity distribution of the particles. Locating and tracking individual particles is often the main objective of many experimental studies in granular flows. If the image quality is good and the scene is not too complex, simple binary analysis or gray level template matching techniques can be used successfully. Unfortunately, many times the image quality may not be good, and the patterns may be only partially visible. An example of closely packed spheres is shown in Figure 1.1, where the objective is to locate as many spheres as possible, with reasonable accuracy. This task is a challenge to many exsiting simple methods, and it requires that the method used is, a) fast since there are many spheres to be detected, and b) reliable for detecting even partially visible spheres.

Generally, 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 [Haralick and Shapiro, 1992]. Image processing and pattern recognition are two closely related computer application areas, which together define the field of computer or machine vision. Image processing is the technique to convert digitally encoded images from a given form into another more useful form. Pattern recognition is the ability to understand structure in sensor-derived data, whether by humans, animals, or computers. Pattern recognition involving image data typically is preceded by various image processing operations to convert the original images into a form that will ease the actual pattern recognition operations. Unlike image processing, pattern recognition is very much context dependent. It is intrinsically a classification task, and can range from simple matching of an unknown pattern against a fixed set of sample patterns to the analysis and matching of an elaborately structured pattern feature set.

Based on the above discussion, we can see that whether it is a robot navigation system, mobile vehicle guidance system, quality inspection system, robot assembly system or tracking systems, environment recognition or object recognition is the common task. This project will focus on object recognition techniques applicable to the above systems.

1.2 The Image Formats

An image is a spatial representation of an object, a two-dimensional or three-dimensional scene, or another image. In computer vision, image usually means recorded image, such as a video image, a digital image, or a picture. It may be abstractly thought of as a continuous function I of two variables defined on some bounded region. The value of the

image located at spatial coordinates *(x,y)* is denoted by *I(x,y)*. For optic or photographic sensors, *I(x,y)* is typically proportional to the radiant energy received in the electromagnetic band to which the sensor or detector is sensitive in a small area around *(x,y)*. In this case the image is called an intensity image. For range finder sensors, *I(x,y)* is a function of the line-of-sight distance from *(x,y)* to an object in the three-dimensional world, and the image is called a range image. For tactile sensors, *I(x,y)* is proportional to the amount by which the surface at and around *(x,y)* deforms the sensor. When the image is a map, *I(x,y)* is an index or a symbol associated with some category such as color, thematic land use, soil type, or rock type. We refer to such an image as a symbolic image [Haralick and Shapiro, 1992].

In computer vision, we usually deal with two kinds of images, intensity images and range images. Based on the above definition, it is not difficult to see that digitized intensity images are arrays of numbers that indicate the brightness at points on a regularly spaced grid and contain no explicit information about depth. Range data are often produced in the form of an array of numbers, referred to as a range image, where the numbers quantify the distances from the sensor focal plane to object surfaces within the field of view along rays emanating from points on a regularly spaced grid. The most attractive feature of range images is the explicitness of the surface information. Many industrial and navigational robotic tasks will be more easily accomplished if such explicit depth information can be efficiently obtained and interpreted. The techniques about intensity images have been widely researched and used in the practical applications. However, most of research about range images is still in the developing stage. This project will focus on development of the

techniques of object recognition from both intensity images and range images, with more emphasis on range images.

1.3 Hough Transform and its Application to Pattern Recognition

The Hough transform [Hough, 1962], HT, is a powerful method of parameter extraction that can convert a difficult global detection problem in image space into a more easily solved local peak detection problem in a parameter space. It has long been recognized as a technique of almost unique promise for shape and motion analysis in images containing noisy, missing, and extraneous data.

The following will illustrate the key ideas of basic HT method by considering extracting a straight line from a set of collinear points in an image. Assume that the image is a two-dimensional one with coordinates (x, y), and the straight line can be expressed by

$$y = mx + c, \tag{1.1}$$

where m and c are two parameters, the slope and intercept, which characterize the line. To determine the equation, m and c need to be decided. Reformat Eq. (1.1),

$$c = -xm + y, \tag{1.2}$$

then we can observe that a point (x_i, y_i) in image space corresponds to a straight line determined by $c = -x_i m + y_i$ in parameter space. All lines corresponding to the points which are collinear in image space should intersect at a common point in parameter space and the coordinates of this common parameter point represent the straight line connecting the image points, i.e., the problem of extracting a straight line from a set of collinear

points in image space is solved by searching a point of intersection in parameter space. Figure 1.2 is a sketch map of the Hough transform for line detection.

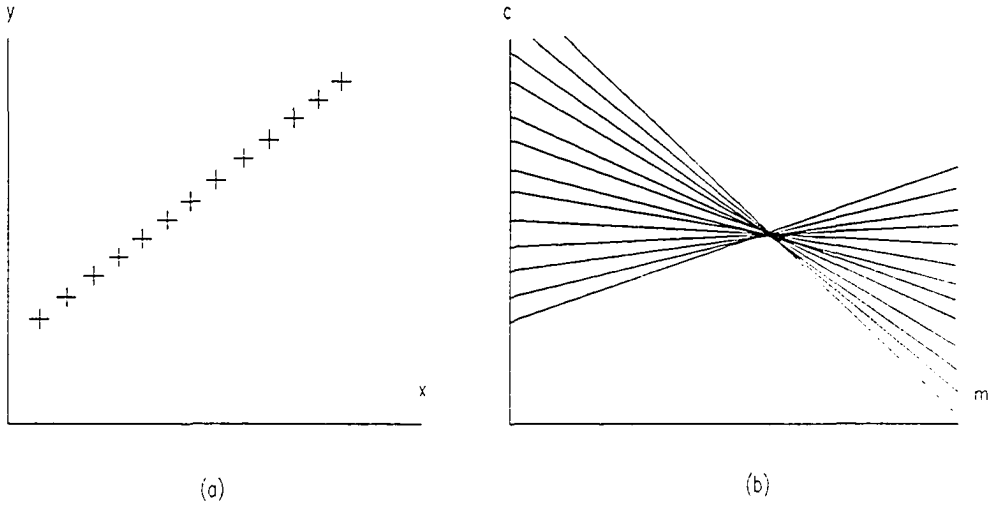


Figure 1.2 The Hough Transform for Line Detection. (a) image space; (b) parameter space.

Actually, the most popular HT method for a straight line is via the equation of the normal vector from the origin to the straight line [Duda and Hart, 1972], i.e.,

$$\rho = x\cos(\theta) + y\sin(\theta), \tag{1.3}$$

where ρ is the length of the vector and θ is the angle it makes with the x axis. Here each point (x_i, y_i) is mapped into a sinusoidal curve in (ρ, θ) parameter space and thus a line is detected by the intersection of these sinusoids.

Based on Hough's ideas, many variations of Hough transform method have been developed [Illingworth and Kittler, 1988], and most recently, some probabilistic approaches have also been developed [Leavers, 1992]. Chapter 2 discusses in greater

detail the merits and demerits of some approaches and proposes a novel probabilistic HT method.

1.4 Fuzzy Clustering Techniques

Fuzzy techniques have been successfully used for a variety of clustering and classification problems. They are based on objective function optimization techniques and generate a fuzzy partition of the data.

Let $\Re$ be the set of real, $V = \{v_1, v_2, ..., v_n\} \subset \Re$ be a set of feature vectors in a feature space, where n is the number of vectors. Let $P = \{p_1, p_2, ..., p_c\}$ represent a c-tuple of prototypes each of which characterizes one of the c clusters. Let u_{ij} be the grade of membership of feature point v_j in cluster p_i, and $U = [u_{ij}]$ is a $c \times n$ matrix called the fuzzy partition matrix satisfying the following conditions:

$$u_{ij} \in [0,1], \qquad i \in \{1,2,...,c\}, j \in \{1,2,...,n\},$$

$$\sum_{i=1}^{c} u_{ij} = 1, \qquad j \in \{1,2,...n\},$$

$$0 < \sum_{j=1}^{n} u_{ij} < n, \qquad i \in \{1,2,...c\}, \tag{1.3}$$

Then the problem of fuzzily partitioning the n feature vectors into c clusters can be formulated as the minimization of an objective function $J(P,U;X)$ of the form

$$J(P,U;V) = \sum_{i=1}^{c} \sum_{j=1}^{n} (u_{ij})^m (d_{ij})^2, \tag{1.4}$$

where, $m \in [1,\infty)$ is a weighting exponent called the fuzzifier, and d_{ij} is the distance from a feature point v_j to the prototype p_i and can be different measurement with different

application. Minimization of the objective function with respect to U subject to the constraints in (1.3) yields [Bezdek, 1981]

$$u_{ij} = \begin{cases} 1/dd & I_j = \varnothing \\ 0 & i \notin I_j, \quad I_j \neq \varnothing \\ 1 & i \in I_j, \quad I_j \neq \varnothing \end{cases} \tag{1.5}$$

where $dd = \sum_{k=1}^{c} (\frac{d_{ij}}{d_{kj}})^{\frac{2}{m-1}}$, $I_j = \{i | 1 \le i \le c, d_{ij}^2 = 0\}$.

With different prototypes P and different definitions of distance d_{ij}, the minimization of the objective function (1.4) leads to different algorithms. Chapter 3 will present more detailed discussion about fuzzy techniques and extend some algorithms to cluster extraction from range images.

1.5 Scope of Research and Outline of Dissertation

In machine manufacturing industry, most parts consist of basic shapes, such as blocks, cylinders, spheres, cones. Developing recognition methods for these basic shapes will greatly contribute to the development of robot navigation system, mobile vehicle guidance system, inspection system, and even robot assembly system in various applications, especially in machine manufacturing environment. Therefore, this project will concentrate on developing object recognition techniques for general shapes and the performance of most algorithms is tested with these basic shapes.

A brief introduction about this project has been presented in this chapter. In what follows, the effort of this research project will be discussed in a greater detail. A survey of Hough transform technique, including recent developments, especially in the aspect of

probabilistic Hough type methods are presented in the next chapter. Some mathematical analyses and investigations of the storage requirements and computation cost of existing Hough transform methods are presented, and a novel probabilistic Hough transform method that is called Fast Randomized Hough Transform (FRHT) is proposed. The FRHT method uses a window based sampling method to randomly pick data points from the data set, which makes the computation speed much faster. The effect of outliers on this method is also analyzed. It can be applied to 2-dimensional intensity images and 3-dimensional range images, and its performance of the method is tested with different data sets.

Chapter 3 deals with fuzzy clustering techniques, including a brief review of fuzzy c-shells type algorithms. Based on the existing fuzzy clustering algorithms, the Fuzzy C Quadric Shells algorithm is extended to the three-dimensional object recognition from range images. Chapter 3 also discusses different distance computation methods that are closely related to the computation speed, convergence and precision of results of fuzzy c shells algorithms.

In Chapter 4 the noise problem in clustering methods is covered. Based on the concept of 'noise prototype' introduced by Dave [Dave 1991], the noise type Fuzzy C Quadric Shells algorithm is developed. It can be used in the object recognition from range images.

Chapter 5 investigates the advantages and disadvantages of Hough transform based methods and fuzzy clustering algorithms, and then proposes a new integrated method combining their advantages. Several examples are given along with the interpretation of results, as well as a comparative study.

Chapter 6 treats validity techniques required for elimination of spurious clusters which is an important step in improving and refining the final clustering results. This chapter gives a brief introduction of current clustering validity measures, and also proposes a new validity criterion for circular clusters.

In Chapter 7 a practical application of object recognition methods to the granular flow research is presented. This research project is supported in part by a U. S. Department of Energy contract.

Chapter 8 provides some theoretical analyses and comparisons of FRHT, noise clustering method, Least Squares (LS) type clustering algorithms, and Least Median Squares (LMS) type clustering methods. Then a new algorithm called Generalized Least Trimmed Squares method for Quadric Surfaces (GLTS-QS) is proposed.

Finally, in Chapter 9, the conclusions of the research are drawn and suggestions are made for the future work.

CHAPTER 2

HOUGH TRANSFORM BASED METHODS

2.1 Introduction

The Hough transform, HT, was first introduced by Paul Hough in a patent filed in 1962. It was used to detect complex patterns of points in binary image data [Hough 1962]. Rosenfeld [1969] brought it to the attention of the image processing researchers, thereafter HT gradually became a popular pattern recognition method. Duda and Hart [1972] used angle-radius (θ,ρ) rather than slope-intercept parameters to parameterize the straight lines. This means that image points map into sinusoidal curves in a two-parameter space, which avoids the singularity problem, limits the parameter space and simplifies the computation. As to the properties of the HT method, much work [Shapiro 1975, 1978a, 1978b, 1978c, 1979, Cohen and Toussaint 1977, Van Veen and Groen 1981, etc.] has been published. Kimme et al [1975] used edge direction information to constrain the range of parameters, then applied the HT technique to detect circular arcs. Several researchers [Wechsler and Sklansky 1977, Tsuji and Matsumoto 1978, Tsukune and Goto 1983, etc.] investigated searching parabolas and ellipses. Merlin and Fraber [1975] presented a generalized HT method to find an arbitrary shape at a given orientation and a given scale. By using directional edge information, Ballard [1981] extended the generalized HT to detect arbitrary shapes for any orientation or any scale. Various results have been successfully achieved by using this idea [Silberberg et al. 1984, 1986, Tsui et al. 1989, etc.].

The key idea of Hough transform, HT, is to convert a difficult global detection problem in image space into a more easily solved local peak detection problem in a parameter space [Illingworth and Kittler, 1988]. The HT has long been recognized as a powerful technique in image processing and data analysis. But, due to its computational and storage complexity, the standard HT has been modified and extended.

The storage or memory in computer is directly related to accuracy of results and the number of parameters to be detected. To achieve better results with the limited memory, one may only focus on smaller meaningful ranges over the large range parameter space. Various approaches have showed that both memory requirements and computational load can be dramatically reduced using an intelligent, iterative, coarse to fine accumulation technique. Li et al. [1986] presented a fast Hough transform method to detect straight lines. The parameter space is partitioned into quadrants and accumulated. Each quadrant is then examined and the algorithm proceeds by subdivision at each scale of those quadrants containing votes in excess of a predetermined threshold. Quadrants are processed until they either fail to produce enough votes or reach a predetermined size. A disadvantage of the method is that the peaks located at the boundaries of quadrants may be missed by the processing. Illingworth and Kittler [1987] developed an adaptive Hough transform method for the detection of 2-D shapes. It uses a small accumulator array and the idea of a flexible iterative "coarse to fine" accumulation and search strategy to identify significant peaks in the Hough parameter spaces. The method is substantially superior to the standard HT implementation in both storage and computational requirements.

However, this method can not detect lines reliably when the number of lines is large and the length of lines is short with respect to the size of the image [Leavers, 1992].

Recently, some probabilistic approaches to the HT have been reported by several researchers. In a simple effort to reduce the amount of computation, a random subsampling of the points in the image space can be used as a reduced but representative data set using which the regular HT method is performed. Kiryati et al. [1991] developed such a method and called a probabilistic Hough transform algorithm for straight line detection. The algorithm randomly samples the image data and only the sampled portion of image points is transformed, which reduces the execution time of the incrementation stage although the results are accumulated in the conventional way. It is shown that the sample size will be directly related to the number of false peaks and the complexity of the image. Since this method only concerns about how to reduce execution time of voting stage, the accuracy of results and the accumulator array remain unchanged.

Better probabilistic approaches that are not just based on a randomly selected reduced data set have been reported. For example, the approach proposed by Califano et al. [1989] where they recognize and make explicit the problems concerning the combinatorial explosiveness inherent in their method. They propose a concept of "generalized neighborhoods." Circular segments are swept out around fixed feature points. Constraints are imposed on the size and location of the segments. Successive segments must overlap and their size is chosen such that the radius of the circle is that distance over which two neighborhoods are expected to contain coherent information. They use a normal n-dimensional accumulator and point out the difficulties in interpreting the

accumulated evidence which are associated with the effects of correlated noise. To find an overall consistent interpretation of the data, they use a "competitive integration" mechanism motivated by connectionist networks. However, the method may not work well for the extraction of long distance correlation when there is not a chain between two distant portions of the same feature.

The randomized Hough transform [Xu *et al.*, 1990, Fu and Dave, 1993], RHT, is a kind of probabilistic method. For the conventional or standard HT, one point in image space is mapped into many points in the parameter space, then the local peaks in the parameter space are searched after all points in image space have been mapped, and the local peaks represent the detected parameters. However, the randomized Hough transform method randomly picks n points in image space for an n-parameter curve, and maps them into one point in the parameter space at each step of the iterative procedure by solving n joint equations. It uses a parameter data set with each element containing both a real valued vector and an integer score to implicitly represent the parameter space, and update the parameter set at each step by the point mapped from the randomly picked points, then searches the parameters of the detected curves from the accumulated parameter data set. For a cluster expressed by an n-parameter linear equation, the general RHT procedure is as following:

(1) Take all points in image space into the image data set D. Then, initialize a parameter data set $P = null$ and set number of iterations $k = 0$.

(2) Randomly pick n points $d_1,...,d_n$ out of D in such a way that all points of D have an equal probability to be taken as d_1, then all points in $D-\{d_1\}$ have an equal

probability to be taken as d_2, etc., finally, all points in $D-\{d_1,d_2,\ldots,d_{n-1}\}$ have an equal probability to be taken as d_n. Substitute values of the n points in the n-parameter equation respectively, then solve the n joint equations to determine one parameter point p_i. Search among set P for an element p_c such that $p_c = p_i$ (or $\|p_c - p_i\| < \delta$, with δ being a given tolerance). If found, go to (4).

(3) Insert p_i into set P as a new element and attach to it an accumulating cell with score one. Go to (5).

(4) Increase the score of the accumulating cell of p_c by one, if the score is not smaller than a given threshold n_t, go to (6).

(5) Accumulate $k = k+1$; if $k > k_{max}$ (k_{max} is a threshold value of iterations), then stop, otherwise, go to (2).

(6) Take p_c as the set of possible parameters of a cluster, if m_p, the number of points lying on this cluster in D is more than m_{min}, the minimum number of points expected in a true cluster, then go to next step; otherwise, take p_c and its accumulating cell out of set P, and go to (2).

(7) Take the m_p points out of D, and a cluster represented by p_c has been detected. Reset $P = null$, $k = 0$, and go to (2).

In the above algorithm, the tolerance δ is introduced to further reduce the storage and adjust the resolution of the parameter space. When $\delta = 0$, the RHT has the highest resolution. The larger δ is, the lower is the resolution but the less storage is used. The threshold k_{max} is used to terminate the processing procedure, i.e., all clusters should be

classified within k_{max} iterations. Obviously, the k_{max} is related with the pixels and pattern of image data.

Compared with the conventional HT, the RHT has some advantages: infinite parameter space, arbitrarily high parameter resolution, small storage, high computation speed. However, in order to effectively apply the RHT to different situations, especially to some more sophisticated cases, such as classifying clusters in range images, we may modify or improve it further with the following aspects:

(1) In RHT, the tolerance δ is introduced to reduce the storage and adjust the resolution of the parameter space. When $\delta = 0$, the RHT has the highest resolution but the most storage which might be impractical and even impossible to accommodate for some kinds of data. When $\delta \neq 0$, the storage is reduced. However when a non-zero value is set for δ, it is difficult to select an appropriate value without any prior information about the image data, since sometimes one value of δ may be too small for one set of data but may be too large for the others.

(2) Further increase detection speed and reduce storage. Though the RHT can effectively increase computation speed and reduce storage memory compared to a conventional HT, there still is a combinatorial explosive problem when the data contains more clusters and each cluster consists of many parameters.

(3) Develop reliable methods to determine whether a point lies on a cluster. This issue is not addressed by Xu *et al.* [1990].

(4) Use a better set of criteria to terminate the searching procedure. Xu *et al.* [1990]

only use a threshold number of iterations to stop the detection, and do not discuss how to appropriately choose the threshold number.

Bergen and Shvaytser [1991] provided a strictly theoretical paper which suggests a Monte-Carlo approximation to the Hough transform. They propose some algorithms which are similar to those of Kiryati et al. [1991] or Xu et al. [1990]. Their analysis deals only with single feature, noiseless data.

Leavers [1992] proposed a dynamic generalized Hough transform method for the concurrent detection of circles and ellipses in 2D images. This method starts from a single connected point which is used to seed the transformation. A segmentation is conducted based on the point and a similar random sampling method as Xu et al. [1990] is used in the segment. As to the memory for an n-parameter shape, the method uses one axis to record the accumulated results of one parameter instead of the n-dimensional space. Hence if T is the resolution in transform space, then the memory requirements will be nT which is much smaller than T^n in the standard Hough transform method. However, this method is not suitable for noisy data like Figure 1.1 and 3D range images because of its segmentation technique, and the claimed memory reduction may not be realized for all kinds of situations due to its single-valued requirement for the transformed parameters.

Based on the above review of recent approaches, a novel probabilistic Hough transform algorithm which can be used in complicated situations, such as feature extraction from range images, will be presented.

2.2 Mathematical Analysis of Hough Transform Methods

Undoubtedly, the HT technique is a powerful tool for pattern recognition and cluster detection. A brief analysis of the technique is presented to gain some insight.

For convenience, assume that there are *noc* clusters within an image without any outliers and *nop* parameters to be decided for each cluster, each cluster contains the same number of points, and the total number of points in the image data is *num*. Then for the conventional Hough transform, a *nop*-dimensional accumulator array is required. Let T be the resolution for all parameters in transform space, then an implementation of the conventional HT requires calculations on the order of

$$N_{HT} = num \times T^{(nop-1)}. \tag{2.1}$$

Actually the conventional HT is a "one to many" mapping method. If we use "many to one" accumulating method, i.e., mapping *nop* points in image space to one point in parameter space by solving *nop* joint equations, then the total number of mappings is

$$N_{MAP} = \binom{num}{nop} = \frac{num!}{nop!(num-nop)!} \tag{2.2}$$

Such "many to one" transform methods can be called "hard mapping" methods.

As to the RHT method [Xu *et al.*, 1990], while all points in the image have an equal probability to be sampled and the number of points is big enough so that the probability of picking a point from any exsiting clusters can be considered as unchanged after some points are picked out, the possible number of mappings required for extracting the first cluster is noc^{nop}, for the second cluster is $(noc-1)^{nop}$, and so on. Then the total possible number of mappings required for extracting all clusters is

$$N_{RHT} = (noc^{nop} + (noc-1)^{nop} + \cdots + 2^{nop} + 1^{nop}) \times n_t = \sum_{i=1}^{noc} (i)^{nop} \times n_t, \qquad (2.3)$$

where n_t is the given threshold number for searching a possible cluster, i.e. when the score at a point or a node of the accumulator reaches n_t the parameters represented by the point are taken as the parameters for a possible cluster and ascertained by further process. The threshold n_t is usually a very small number, like 2 or 3.

By observing the above three equations for the three parameters, *num*, *nop* and *noc*, which have direct effect to computation speed, we can see that N_{HT} and N_{MAP} are independent of *noc*, whereas N_{RHT} is independent of *num*. Eq. (2.1) shows us that, with T and *nop* increasing, N_{HT} will increase significantly. In addition, the domain of Hough space and its resolution T have to be decided before the mapping action, which may restrict the application of HT. Hence, the conventional HT may not be applied to a complicated case which either needs high dimensional accumulator and high resolution or it is difficult to decide the domain of the transform space. Obviously, it may also not be a good method for dealing with the data set containing a large number of points. For the "hard mapping" method, from the expression of N_{MAP}, it is not difficult to see that there is a combinatorial explosive problem. Perhaps that is why there has not been any application of it after Hough proposed his transform method in 1962. Eq. (2.3) tells us that comparing with the other two methods, N_{RHT} will be much smaller since it is only related with the small numbers, *nop*, *noc*, and n_t. That is the main advantage of this method. Before drawing the conclusion, let us use Eqs. (2.1), (2.2) and (2.3) to create the following tables for the comparison among the different methods. For convenience, set

$T = 100$, $n_t = 2$, and $num = 2000$. In those tables, all the numbers larger than 10^6 are rounded. It is necessary to point out that the time for each mapping in RHT and "hard mapping" is usually one or two orders more than the time in HT. However, the total computing time by RHT is much shorter than the time by HT because the large difference of number of mappings between them.

Table 2.1 Computation Load Comparison Among Different Methods with $noc = 1$.

nop	2	3	4	5	6	7	8	9	10
N_{HT}	2.0E+05	2.0E+07	2.0E+09	2.0E+11	2.0E+13	2.0E+15	2.0E+17	2.0E+19	2.0E+21
N_{MAP}	2.0E+06	1.3E+09	6.6E+11	2.7E+14	8.8E+16	2.5E+19	6.3E+21	1.4E+24	2.8E+26
N_{RHT}	2	2	2	2	2	2	2	2	2

Table 2.2 Computation Load Comparison Among Different Methods with $noc = 2$.

nop	2	3	4	5	6	7	8	9	10
N_{HT}	2.0E+05	2.0E+07	2.0E+09	2.0E+11	2.0E+13	2.0E+15	2.0E+17	2.0E+19	2.0E+21
N_{MAP}	2.0E+06	1.3E+09	6.6E+11	2.7E+14	8.8E+16	2.5E+19	6.3E+21	1.4E+24	2.8E+26
N_{RHT}	10	18	34	66	130	258	514	1026	2050

Table 2.3 Computation Load Comparison Among Different Methods with $noc = 5$.

nop	2	3	4	5	6	7	8	9	10
N_{HT}	2.0E+05	2.0E+07	2.0E+09	2.0E+11	2.0E+13	2.0E+15	2.0E+17	2.0E+19	2.0E+21
N_{MAP}	2.0E+06	1.3E+09	6.6E+11	2.7E+14	8.8E+16	2.5E+19	6.3E+21	1.4E+24	2.8E+26
N_{RHT}	110	450	1958	8850	41030	193650	925958	4.5E+06	2.2E+07

Table 2.4 Computation load comparison among different methods with $noc = 10$.

nop	2	3	4	5	6	7	8	9	10
N_{HT}	2.0E+05	2.0E+07	2.0E+09	2.0E+11	2.0E+13	2.0E+15	2.0E+17	2.0E+19	2.0E+21
N_{MAP}	2.0E+06	1.3E+09	6.6E+11	2.7E+14	8.8E+16	2.5E+19	6.3E+21	1.4E+24	2.8E+26
N_{RHT}	770	6050	50666	441650	4.0E+06	3.6E+07	3.4E+08	3.1E+09	3.0E+10

The numbers in the tables clearly show us that the computation load for all three methods increases with the increment of *nop* while *num* and *noc* are fixed; the RHT

method has the least computation in any cases, the standard HT is the next, whereas the "hard mapping" method has the slowest computation speed. Since the computation load of the RHT method is independent of the number of points which is usually much bigger than other parameters, the RHT particularly demonstrates its advantages while dealing with the images containing huge amount of data. Moreover, even if n_t is increased by an order of magnitude, the N_{RHT} will go up only by a factor of 10.

For the Hough type transform method, the computation cost and the accuracy of result are two important aspects to be concerned. While some analysis to the computation has been presented as above, the following will investigate the level of accuracy.

As we know, we usually have to determine the resolution T and the boundary of transform space before using the conventional HT. Assume that the upper bound and lower bound are B_u and B_l respectively for each parameter. Then the accuracy of result A will be

$$A = \frac{B_u - B_l}{T}. \tag{2.4}$$

To achieve higher accuracy, we can take either bigger T or smaller $B_u - B_l$. However, the $B_u - B_l$ is limited by the feature of image data, and the T is restricted by the memory of computer. For sake of simplicity, T, B_u and B_l are assumed to be same for all parameters. For an nop-dimensional problem, the memory requirement M_{HT} can be approximated as

$$M_{HT} = T^{nop}, \tag{2.5}$$

i.e., T^{nop} cells of accumulator array is required for storing results in the transform space. For a high dimensional image, for example, it is impossible to use a higher resolution T

due to limited memory in most present computers. For example, if $T = 100$, and $nop = 9$, then an array of size 100^9 or 10^{18} has to be defined. If the array is an integer one, the memory requirement will be 4.0E+13 mega-bytes on a 32-bit computer.

In reality, all of the predefined parameter space in the conventional HT method is rarely completely used. This phenomena has attracted many researchers and led to different novel methods, such as the early mentioned adaptive HT and various probabilistic HT methods. The adaptive Hough transform method [Illingworth and Kittler, 1987] uses a small accumulator array and the idea of a flexible iterative "coarse to fine" accumulation and search strategy to identify significant peaks in the Hough parameter spaces. For the same example as above, the same accuracy may be achieved with a smaller array here. Using $T = 10$, then an array of size 10^9 is required. The adaptive method uses this array to get some coarse peaks first, then iteratively applies the same accumulating method to the areas around those coarse peaks separately until the fine results are achieved. The method can use much less memory to achieve results with the same accuracy in comparison with the standard HT method. But the number like 10^9 is still too big for most computers as it would require 4 Gbytes or more. In other words, the method is not suitable for the images with high dimensional parameters, such as the range images containing quadric curved surfaces. The RHT method [Xu et al., 1990] uses a parameter data set with each element containing both a real valued vector and an integer score to implicitly represent the parameter space. The dynamic data structure is adopted to store the parameter data set and only the valid points in the parameter space are stored. For a multiple-cluster image, i.e., $noc \geq 2$, the parameter space is reset as *null* as soon as a new cluster is detected. So

the method will not waste any memory space. From Eq. (2.3), it is easy to derive the expression for possible number of elements in the parameter space, M_{RHT}, as

$$M_{RHT} = noc^{nop}. \tag{2.6}$$

The only difference between Eqs. (2.5) and (2.6) is the root of exponential function. Generally, T is much bigger than noc, so M_{HT} is much much bigger than M_{RHT}, for example, when $T = 100$, $noc = 10$, $nop = 9$, then $M_{HT} = 10^{18}$, $M_{RHT} = 10^{9}$. Although the RHT needs more memory for each element of parameter data set than the conventional HT, that difference is not significant in comparison with the difference between M_{HT} and M_{RHT}. Furthermore, the RHT introduces a tolerance δ to further reduce the memory requirements. Therefore, it is not difficult to conclude that, the RHT method needs less memory than the standard HT for most images, especially for the images containing clusters with more parameters.

As mentioned before, theoretically the RHT possesses the highest level of accuracy when $\delta = 0$. By adjusting δ, the different level of accuracy can be achieved.

2.3 The Fast Randomized Hough Transform Method

The review and analysis of Hough techniques has been presented in the two preceding sections. Generally, the RHT method is an outstanding one among various efforts, which possesses some good features such as high computation speed, less storage requirement, and high accuracy of results. However, some improvements are needed in order to apply the method to the more complicated cases, like range images containing more clusters with more parameters, for instance, if $noc = 10$, $nop = 9$, then the number of mappings

$N_{RHT} \cong 3.1 \times 10^9$, and the possible number of elements in the parameter space $M_{RHT} = 10^9$, which obviously is a challenge to most computers.

The number of mappings reflects the computation load, and the possible number of elements in the parameter space is directly related to the required volume of storage. Reducing N_{RHT} and M_{RHT} will increase computation speed and reduce memory requirements. From Eqs. (2.3) and (2.6), it seems impossible to change N_{RHT} and M_{RHT} as the two related parameters, *noc* and *nop* are intrinsic to an image. But, carefully checking the Table 2.1 again, where $noc = 1$, we can find an interesting fact that N_{RHT} takes a small constant 2 independent of the *nop*. That means that the computation load will be light and the storage will be small whenever the number of clusters is small. Though it is impossible to change the inherent number of clusters in an image, there is a practical method that can reduce the number of clusters within a local region by dividing the whole image area into multiple regions, which is the key idea of the following fast randomized Hough transform method (FRHT).

Before describing the new method, some analysis regarding the computation is required. For the sake of simplified analysis, assume that the image contains *noc* clusters without outliers, each cluster consists of *nop* parameters and contains the same number of data points, and the total number of points in the image is *num*. If the image is divided into *nor* different regions where each region equally contains *rc* clusters, then

$$noc = nor \times rc, \tag{2.7}$$

the number of mappings with the fast randomized Hough transform,

$$N_{FRHT} = nor \times \sum_{i=1}^{rc} i^{nop} \times n_t, \tag{2.8}$$

and the possible number of elements in the parameter space,

$$M_{FRHT} = rc^{nop}. \tag{2.9}$$

To see the effect of FRHT, two tables are presented, Tables 2.5 and 2.6, with different rc under the following initial conditions: $noc = 10$, $n_t = 2$.

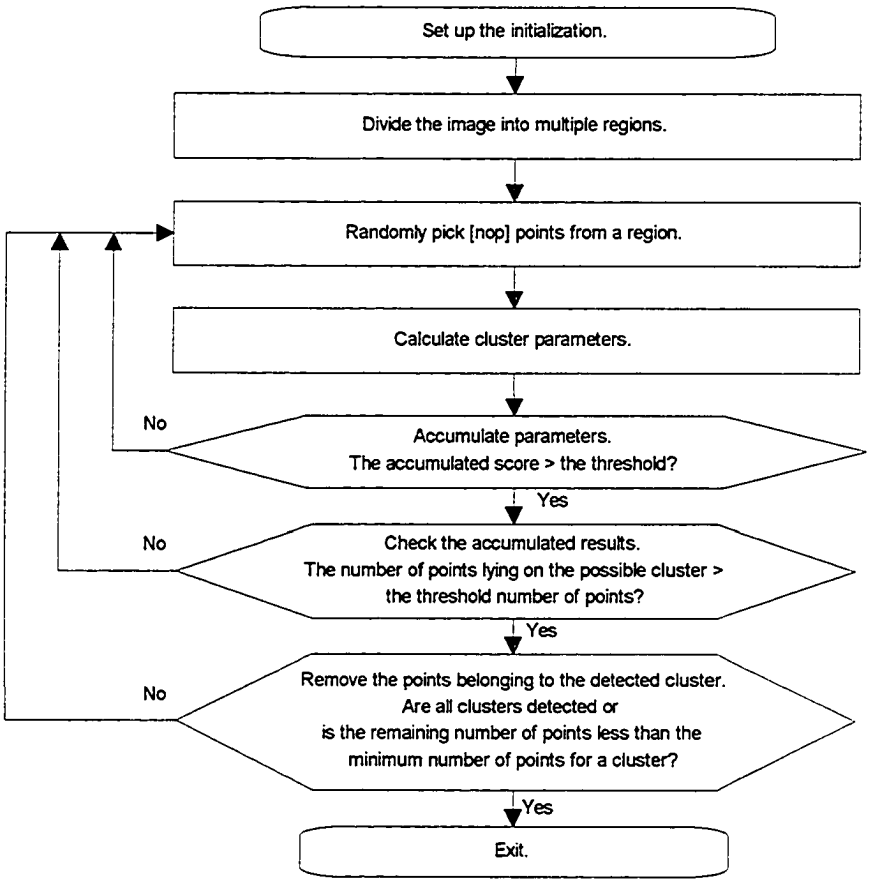


Figure 2.1 Flow Diagram of Fast Randomized Hough Transform.

The two comparison tables clearly indicate that the new Hough transform method can greatly increase the computation speed and reduce the storage memory, which is why it is called the 'fast randomized Hough transform' method. A number of assumption were made to arrive at the number in these tables, and their implication will be discussed later. The following deals with the details of the FRHT method. A flow diagram of the method is shown in Figure 2.1 and all the general information and particular details about it will follow. Since the range images are a challenge to many existing algorithms and are also a focus of this project, it is assumed that the image in question is a range image in the following description except where indicated.

Table 2.5 Computation and Storage Comparison Between RHT and FRHT with $rc = 1$.

nop	2	3	4	5	6	7	8	9	10
N_{RHT}	770	6050	50666	441650	4.0E+06	3.6E+07	3.4E+08	3.1E+09	3.0E+10
N_{FRHT}	20	20	20	20	20	20	20	20	20
M_{RHT}	100	1000	10000	100000	1.0E+06	1.0E+07	1.0E+08	1.0E+09	1.0E+10
M_{FRHT}	1	1	1	1	1	1	1	1	1

Table 2.6 Computation and Storage Comparison Between RHT and FRHT with $rc = 2$.

nop	2	3	4	5	6	7	8	9	10
N_{RHT}	770	6050	50666	441650	4.0E+06	3.6E+07	3.4E+08	3.1E+09	3.0E+10
N_{FRHT}	50	90	170	330	650	1290	2570	5130	10250
M_{RHT}	100	1000	10000	100000	1.0E+06	1.0E+07	1.0E+08	1.0E+09	1.0E+10
M_{FRHT}	4	8	16	32	64	128	256	512	1024

2.3.1 Initialization and Dividing an Image

Generally, a range image contains many more points than for example the edge map of an intensity image does. The range data represents the surface feature of objects. Besl and

Jain [1986] presented the invariant surface characteristics in range images and classified the image regions into eight surface types by the mean curvature and Gaussian curvature. Here, for a general quadric curved surface, the model can be expressed as

$$a_1x^2 + a_2y^2 + a_3z^2 + a_4xy + a_5xz + a_6yz + a_7x + a_8y + a_9z + a_{10} = 0. \qquad (2.10)$$

Obviously, there are ten parameters for each surface, and nine of them are independent. When some of the parameters are set to zero, Eq. (2.10) represents a specific case, for instance, if $a_1 = a_2 = a_3 > 0, a_{10} < 0$ and $a_4 = a_5 = a_6 = a_7 = a_8 = a_9 = 0$, then the surface represents a sphere. Without any prior information about the detected surface, a general equation of (2.10) is always taken as the default model.

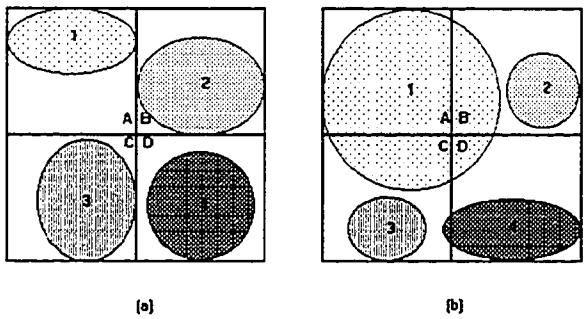


Figure 2.2 Dividing an Image into Multiple Regions.

To make the dividing procedure simple, it is suggested to divide the whole image into multiple regions such that each one has approximately the same area. The number of regions depends on the minimum number of points a cluster must contain. The previous

analysis indicates that, in order to achieve an optimum increase in computation speed and reduction in storage memory, each region should contain only one cluster. Unfortunately, it is a tough requirement for most practical applications, where each cluster may contain different number of points and occupy irregular areas. Figure 2.2 displays the top view of different hypothetical cases of an image, where numbers indicate different clusters and letters represent the different regions.

Figure 2.2a is an ideal case in which each region contains one cluster. In Figure 2.2b, cluster No. 1 covers four regions, whereas cluster No. 4 covers two regions. If the searching sequence for all regions is A-B-C-D, then the cluster No. 1 is identified and removed first, the next will be No. 2 and No. 3. However, since some points of cluster No. 4 are located in region C, there are still some data points in the region after No. 3 is removed. In this case, it is recommended to go to the next region because there are not enough points left in region C, which brings up a criterion for stopping the search within a region, i.e., when a region does not contain enough number of points, it will not be searched further. Usually the threshold number of points is same as the minimum number of points for a cluster. If the searching sequence is D-C-B-A, the cluster No. 4 will be detected first and the data points of cluster No. 1 in region D will be ignored according to the above discussed criterion. While searching region C, the data points of cluster No. 1 in region C should not be ignored since they are comparable with the data points in cluster No. 3. Either No. 3 or No. 1 may be detected first. Therefore, to avoid missing a cluster, it is necessary to set the maximum number of clusters within a region as two or more. However, to give consideration to the computation speed and storage, the number should

not be too big. To put together the above considerations, another criterion for stopping the search within a region may be imposed, namely, when two clusters have been identified within one region, the searching procedure is moved to the next region. Consequently, according to Eq. (2.8), N_R, the maximum number of mappings for a region can be decided by

$$N_R = (2^{nop} + 1) \times n_t. \tag{2.11}$$

It has to be pointed out that the number determined by Equation (2.11) can only be taken as a reference, and the actual maximum number of mappings for a region might need a larger one due to actual situations in applications, for example, some regions might contain more than two clusters.

In the above discussion, all regions are searched sequentially according to the physical locations, i.e. row by row or column by column. Equation (2.8) is derived with the assumption that each cluster contains same number of points. Actually in most applications, different cluster may contain different number of points. Consequently, the searching sequence will affect the total number of distance computations, namely, when the clusters containing more points are detected earlier, the total number of distance computations is smaller. To keep a fair computation load, the FRHT should search the regions with more points earlier. For instance, in Figure 2.2b, the searching sequence should be A-D-B-C. Though this kind of searching sequence can not guarantee that the cluster with greatest number of points would be detected at the earliest searching, the clusters with more points could be detected earlier.

In the conventional HT, an accumulator needs to be set in advance. In the FRHT, a storage structure should also be arranged in advance. Because a new point needs to be inserted to an appropriate location in the parameter space when it is obtained, a binary tree data structure is suggested to use here. The binary tree data structure has proved to be a fast technique to perform searching, inserting and erasing operations [H. Schildt, 1988]. Hence only a binary tree needs to be defined in advance instead of an array. To sum up the above discussion, the initialization consists of the following:

(1) Divide the image into multiple regions;

(2) Set up the model of the clusters;

(3) Select the minimum number of points a cluster must contain;

(4) Choose the maximum number of clusters to be searched in a region as 2;

(5) Determine the maximum number of mappings in a region;

(6) Decide the threshold number of points for searching a region, which is usually same as the minimum number of points for a cluster;

(7) Build a binary tree for storing the accumulating results and set the root of binary tree as *null*.

2.3.2 Sampling of Data Points

After an image is divided into multiple regions, the FRHT will search clusters through all those regions one by one. In the searching procedure, a random sampling method is adopted to pick *nop* different points from a region, where *nop* is the number of independent parameters, as an example, $nop = 9$ for the model with the expression of

(2.10). During the sampling of data points, it is required that all points in the sampling region have an equal probability to be taken as one of the *nop* points, and any one point in the region can be picked only one time for each sampling. Here a sampling is defined as the random sampling of *nop* points. Obviously, one sampling corresponds to one mappings. If the number of sampling reaches the maximum number of mappings in a region, the searching within the present region will be terminated.

The previous mathematical analysis shows that the random sampling method may greatly increase the detection speed and save storage space.

2.3.3 Calculating the Parameters

The FRHT is a kind of "many to one" transform method, which maps *nop* points in image space into one point in parameter space by solving *nop* joint equations. In other words, *nop* joint equations can be obtained by substituting the values of randomly picked points into the model expression, and consequently, one point in parameter space, i.e., *nop* parameters, can be computed by solving those joint equations.

2.3.4 Accumulating the Parameters

After the values of a point in parameter space $C = \{c_1, c_2, \ldots, c_{nop}\}$ are obtained, unlike the RHT method, they are normalized and then inserted into the data tree. The normalization is done by dividing all elements in C by the one with biggest absolute value, i.e.,

$$\tilde{C} = \frac{C}{Max\{|c_1|, |c_2|, \ldots, |c_{nop}|\}}, \tag{2.12}$$

where $\tilde{C}$ represents the normalized values of C. All the normalized values will be restricted as

$$0 \le |\tilde{c}_i| \le 1 \quad , \quad i \in \{1,2,\ldots,nop\}, \tag{2.13}$$

where $\tilde{c}_i$ is the normalized value of c_i. Thereafter, if there is a node in the binary tree with the same normalized values, the score of the node will be increased by one, otherwise, those values will be stored at a proper accumulating cell with score one. Theoretically, two sets of values may be the same, but in practice even the two sets of values from one cluster may have some bias due to the computation accuracy in the computing device. Therefore, it is necessary to give a tolerance δ for the values at a node, by which it can be decided if two sets of values can be merged or not. Therefore, if set $\tilde{c}'_i$ as the previous set of values at a node and

$$|\tilde{c}_i - \tilde{c}'_i| < \delta \, , \quad i \in \{1,2,\ldots,nop\}, \tag{2.14}$$

then consider the two sets of values as the same. It is easy to see that δ can be taken as the resolution of parameters. Furthermore, because all the values in parameter space are normalized, the magnitude of δ represents the percentage error of results and has a practical meaning for any kind of image data, and it can also be easily predetermined even in absence of prior information.

During the accumulation, if the score at a point has reached to a given number n_t, then the points may correspond to a cluster and the parameter will be examined in a way discussed in the following section. The given number n_t depends on image quality, and is usually suggested to take a small number, like 2 or 3, but if the data contains too much noise, a bigger number is recommended. It is advised to take the number as small as

possible since it affects the computation speed, which can be figured out from the previous mathematical analysis.

2.3.5 Inspection of the Results

When the score at a point in parameter space reaches n_t, the parameters at the point may represent a cluster. The FRHT takes $\tilde{C}$ as the possible parameters of a cluster and checks how many points in the whole image lie on the cluster. If the number of points is more than the defined minimum number of points in a cluster, $\tilde{C}$ represents a true cluster, all points belonging to the cluster is removed from all the divided regions, and the root of the binary tree is reset as *null*, otherwise, $\tilde{C}$ represents a false cluster and is erased from the binary tree.

How to determine a point lies on a cluster may appear to be simple as it could be decided just by checking whether the values of the point satisfy the expression of the detected cluster. In reality, it is difficult to satisfy the expression even with the points on the cluster because of different types of errors, such as the errors from data acquisition and computation. This problem may be solved by setting a tolerance ε, i.e., if the values of a point (x_i, y_i, z_i) are substituted into the expression of the cluster and satisfy

$$\left\| a_1 x_i^2 + a_2 y_i^2 + a_3 z_i^2 + a_4 x_i y_i + a_5 x_i z_i + a_6 y_i z_i + a_7 x_i + a_8 y_i + a_9 z_i + a_{10} \right\| < \varepsilon, \quad (2.15)$$

then the point can be considered to lie on the cluster. However, a proper value for ε can not be easily selected without prior information about the cluster as it does not have any physical meaning. Therefore, it is suggested to use the actual distance between a point and a cluster to determine whether a point belongs to a cluster. The distance D is defined as

the shortest distance between the point (x_i, y_i, z_i) and the cluster. Let (x_c, y_c, z_c) be the point on the cluster, which has the shortest distance to point (x_i, y_i, z_i), then

$$D = \min(\sqrt{(x_i - x_c)^2 + (y_i - y_c)^2 + (z_i - z_c)^2}), \text{ subject to}$$

$$a_1 x_c^2 + a_2 y_c^2 + a_3 z_c^2 + a_4 x_c y_c + a_5 x_c z_c + a_6 y_c z_c + a_7 x_c + a_8 y_c + a_9 z_c + a_{10} = 0. \quad (2.16)$$

The Lagrangian method and the routines from MINPACK (refer to the Appendix A for more details) can be used to calculate this distance. In Equation (2.16), nine of ten parameters of the model are independent and the other one can be set as a constant in actual applications. With the Euclidean distance, it is easy to set a threshold value D_t to classify a point to a cluster. If the distance D between a point and a cluster satisfies $D < D_t$, then the point is classified to the cluster.

2.4 The Effect of Outliers on the Fast Randomized Hough Transform

The preceding sections present the mathematical analysis with the assumption that the image does not contain any outliers, i.e., every point in the image can be classified to a cluster. This section will discuss the effect of outliers on the fast randomized Hough transform method.

Let us assume that there are *noc* clusters within an image having outliers and *nop* parameters to be decided for each cluster. From the previous analysis of RHT and FRHT, we know that the possible number of mappings is independent of the total number of points, so the percentage is used here to indicate the amount of outliers. Let β_e be the percentage amount of outliers in the image, and β_i be the percentage in ith cluster, then

$$\beta_1 + \beta_2 + \cdots + \beta_{noc} + \beta_e = 1. \quad (2.17)$$

Usually, the cluster with more points is detected earlier than other clusters with less points, hence, without loss of generality, we define

$$\beta_1 \geq \beta_2 \geq \cdots \geq \beta_{noc}. \tag{2.18}$$

Similarily as Eq. (2.3), the possible number of mappings for extracting all clusters by the RHT method is

$$N'_{RHT} = (\frac{1}{\beta_1})^{nop} \times n_t + (\frac{1-\beta_1}{\beta_2})^{nop} \times n_t + \cdots + (\frac{1-\beta_1-\beta_2-\cdots-\beta_{noc-1}}{\beta_{noc}})^{nop} \times n_t,$$

or

$$N'_{RHT} = \sum_{i=1}^{noc} (\frac{1-\sum_{j=0}^{i-1}\beta_j}{\beta_i})^{nop} \times n_t, \tag{2.19}$$

where $\beta_0 = 0$. The expression for possible number of elements required in the parameter space becomes

$$M'_{RHT} = (\frac{1}{\beta_1})^{nop}. \tag{2.20}$$

Obviously, Eq. (2.3) and (2.6) are the special cases of (2.19) and (2.20) respectively, where $\beta_1 = \beta_2 = \cdots = \beta_{noc}$ and $\beta_e = 0$. To see the effect of outliers on the RHT method, set $n_t = 2$, $noc = 10$, and $\beta_1 = \beta_2 = \cdots = \beta_{noc}$, then two tables with different amount of outliers and different number of parameters are created as Tables 2.7 and 2.8.

As to the FRHT method with outliers, in order to simplify the analysis, assume that the image is divided into *nor* different regions where each region contains same percentage of outliers and same number of clusters, *rc*, and each cluster has the same amount of points, β, then Eq. (2.7) still helds true here, Eq. (2.17) becomes

$$noc \times \beta + \beta_e = 1, \tag{2.21}$$

and the possible number of mappings for extracting all clusters with the fast randomized Hough transform,

$$N'_{FRHT} = nor \times \sum_{i=1}^{rc} \left(\frac{1-(i-1)\times nor \times \beta}{nor \times \beta}\right)^{nop} \times n_t. \tag{2.22}$$

Correspondingly, the possible number of elements required in the parameter space,

$$M'_{FRHT} = \left(\frac{1}{nor \times \beta}\right)^{nop}. \tag{2.23}$$

Table 2.7 Computation Load with Different Amount of Outliers β_e.

nop	2	3	4	5	6	7	8	9	10
$\beta = 0\%$	770	6050	50666	441650	4.0E+06	3.6E+07	3.4E+08	3.1E+09	3.0E+10
$\beta = 10\%$	1039	9052	83893	809254	8.0E+06	8.1E+07	8.3E+08	8.7E+09	9.1E+10
$\beta = 20\%$	1445	14200	147697	1.6E+06	1.8E+07	2.0E+08	2.3E+09	2.7E+10	3.2E+11
$\beta = 50\%$	4970	82150	1.4E+06	2.4E+07	4.3E+08	7.7E+09	1.4E+11	2.6E+12	4.8E+13

Table 2.8 Storage Requirements with Different Amount of Outliers β_e.

nop	2	3	4	5	6	7	8	9	10
$\beta = 0\%$	100	1000	10000	100000	1.0E+06	1.0E+07	1.0E+08	1.0E+09	1.0E+10
$\beta = 10\%$	123	1372	15242	169351	1.9E+06	2.1E+07	2.3E+08	2.6E+09	2.9E+10
$\beta = 20\%$	156	1953	24414	305176	3.8E+06	4.8E+07	6.0E+08	7.5E+09	9.3E+10
$\beta = 50\%$	400	8000	160000	3.2E+06	6.4E+07	1.3E+09	2.6E+10	5.1E+11	1.0E+13

To see the effect of outliers on FRHT, the following tables, 2.9 to 2.12, are established with different *rc* and different β_e under the following initial conditions: $noc = 10$, $n_t = 2$.

The numbers in these tables show that for the RHT and FRHT methods, with the amount of outliers inreasing, the computation speed will decrease and the storage requirements will increase. This can be easily understood as that for the whole image with the RHT or a region with the FRHT method, the outliers or noisy points can be taken as

an extra cluster which can not fit a model, and the total number of clusters in the searching region increases by one, therefore the possible number of mappings and the possible storage requirements increase.

Table 2.9 Computation Load with Different Amount of Outliers β_e and $rc = 1$.

nop	2	3	4	5	6	7	8	9	10
$\beta = 0\%$	20	20	20	20	20	20	20	20	20
$\beta = 10\%$	25	27	30	34	38	42	46	52	57
$\beta = 20\%$	31	39	49	61	76	95	119	149	186
$\beta = 50\%$	80	160	320	640	1280	2560	5120	10240	20480

Table 2.10 Computation Load with Different Amount of Outliers β_e and $rc = 2$.

nop	2	3	4	5	6	7	8	9	10
$\beta = 0\%$	50	90	170	330	650	1290	2570	5130	10250
$\beta = 10\%$	64	128	266	569	1238	2717	5997	13277	29442
$\beta = 20\%$	85	190	441	1053	2555	6274	15515	38531	95944
$\beta = 50\%$	250	910	3370	12670	48250	185710	720970	2.8E+06	1.1E+07

Table 2.11 Storage Requirements with Different Amount of Outliers β_e and $rc = 1$.

nop	2	3	4	5	6	7	8	9	10
$\beta = 0\%$	1	1	1	1	1	1	1	1	1
$\beta = 10\%$	1.2	1.4	1.5	1.7	1.9	2.1	2.3	2.6	2.9
$\beta = 20\%$	1.6	2.0	2.4	3.1	3.8	4.8	6.0	7.5	9.3
$\beta = 50\%$	4	8	16	32	64	128	256	512	1024

Table 2.12 Storage Requirements with Different Amount of Outliers β_e and $rc = 2$.

nop	2	3	4	5	6	7	8	9	10
$\beta = 0\%$	4	8	16	32	64	128	256	512	1024
$\beta = 10\%$	5	11	24	54	120	268	595	1322	2937
$\beta = 20\%$	6	16	39	98	244	610	1526	3815	9537
$\beta = 50\%$	16	64	256	1024	4096	16384	65536	262144	1.0E+06

Although the outliers affect the cluster detection with the FRHT method, the previously described steps are still applicable to the images with outliers or noisy points

except that the maximum number of mappings in a region needs to be set to a bigger value.

2.5 Numerical Results and Conclusions

The FRHT method is described in above sections, in what follows, some experimental results and conclusions will be presented.

To test the performance of the FRHT algorithm, a range image with four different ten-parameter quadric surfaces is created as shown in Figure C.1 (see Appendix C) that contains 19600 data points, and all original parameters for each surface cluster are listed in Table 2.13. As discussed before, the model equations for all the clusters can be represented as Equation (2.10), which contains nine independent parameters. Obviously, the conventional Hough transform method can not be used to solve this problem because of the huge memory requirements. Set the minimum number of points for a cluster as about 10% of the total number of points, the accumulating threshold number n_t as 2, the tolerence δ as 0.0001, and the threshold distance D_t as 0.05, then the detection result by the RHT method is shown in Figure C.2 and the parameters are listed in Table 2.14. The CPU time for the detection on a SPARC 10 Station is 354.6 seconds. Dividing the image into 3×2 regions and using the same initial conditions, the detection result by the FRHT is shown in Figure C.3 and the detected parameters are listed in Table 2.15. The CPU time on a SPARC 10 Station is 230.2 seconds that is less than the time used by the RHT method.

Based on the image in Figure C.1, two noise images in Figures C.4 and C.5 are generated in such a way that 1960 and 3920 noise/outlier points, which are about the same

as 10% and 20% of the total number of points in the original image, are randomly added to it respectively. With the same initial conditions as above examples, the detection results by the RHT and the FRHT methods are shown as Figures C.6 to C.9, and the detected parameters are listed in Tables 2.16 to 2.19. The CPU time used on a SPARC 10 Station by different methods is listed in Table 2.20. The results indicate that the FRHT method possesses faster computation speed than the RHT method, and the image with more noise points takes more detection time. It has to be pointed out that because the images used here contain huge amount of data points, and since the Euclidean distance computation for all points is very time consuming, much of the CPU time in the above experiments is spent for the distance computation, and the advantage of the FRHT method is not that obvious. For a simplified image as shown in Figure C.10, which is generated with the same quadric surface parameters as the image in Figure C.1 but only about 1600 data points, the RHT method obtains the parameter results shown in Table 2.21 in 140.0 seconds, whereas the FRHT method uses 17.6 seconds to obtain the results as shown in Table 2.22. Certainly, while detecting the simplified image, the minimum number of points for a cluster needs to be set as a smaller number, like 15% of the total number of points. Now we can see that the computation speed with the FRHT method is much faster than that with the RHT method. It is worthy to point out that for the huge images, it is strongly recommended to randomly sample the data and then apply FRHT with Euciliden distance computations, which will greatly reduce the computation load.

The above noise images are generated by adding some random noise points to a non-noise image thus amounting to adding outliers. However, the noise can also be

generated by randomly shifting the original points along a direction within a specific distance. The latter case is more like what happens in the practical applications. A noise image generated by randomly shifting all points in Figure C.10 along z direction within ±0.05 unit distance is shown as Figure C.11. Here we set the minimum number of points for a cluster as 15% of the total number of points, the accumulating threshold number n_t as 2, the tolerence δ as 0.001, and the threshold distance D_t as 0.2, and apply the FRHT method to the noise image, then the graph result is shown in Figures C.12 and the parameter result is shown in Tables 2.23. The CPU time on a SPARC 10 Station is 146.2 seconds. But with the same initial conditions, the RHT method is not able to identify the clusters in the image.

As mentioned above, the Eucilidean distance computations are very time consuming that might not be practical for some applications. To speed up the computation, some methods of approximate distance computation have been proposed [Taubin 1991, Haralick and Shapiro 1992], which will be discussed in greater detail in Chapter 3. Using the first-order distance approximation method [Haralick and Shapiro 1992], the FRHT algorithm and its mathematical analyses can be interpreted more completely. With the approximate distance computations, the results for the image in Figure C.1 by RHT and FRHT methods are shown in Table 2.24, where different perentage amount, from 20% to 100%, of data points are randomly sampled before the algorithms are used, and the CPU time is clocked on a SPARC 10 Station. In Table 2.24, N_{theo} represents the theorotically estimated number of samplings assuming each cluster contains same number of points, whereas N_{actual} represents the actual number of

samplings. From the table, we can see that the actual number of samplings is independent of the total number of points, and the computation speed with RHT is much slower than the speed with FRHT. While the RHT method is used, the CPU time mostly depends on the actual number of samplings that is usually very big, and while the FRHT method is used, the CPU time depends on the total number of points in the image as the actual number of samplings is very small. Through the experiment, it was found that most of the CPU time used by FRHT is to read and sample the data at the beginning, for instance, in the above example, 11.5 seconds was used to read and sample the data, which means that the actual computation time by FRHT could reach to a level of one tenth of a second.

Another example is considered as shown in Figure C.13, which is an image of 200x200 with three clusters, one sphere, one cylinder and one frustum of a cone, artificially generated on the computer. There are 4062 points in the image and the depth values are recorded in floating numbers. All three clusters can be correctly detected by the FRHT method with 3x3 windows in 5.0 seconds on a Sparc 10 computer, and the results are shown in Figure C.14.

Based on the analysis and the experiments, we can conclude that the FRHT method possesses the following characteristics: high computation speed, low memory requirement, high result resolution and infinite parameter space. It can be used to detect complex shapes, such as the quadric surface image containing several clusters and noisy points. Through the experiments, it was noticed that sometimes for noise images, the FRHT method is sensitive to the number of regions, in other words, it does not work well

for some noise images. In Chapter 5, this issue will be further discussed and strategies to improve the performance of the FRHT method will be presented.

Table 2.13 The Original Parameters for Creating the Four Clusters in the Range Image as Shown in Figure C.1.

Cluster	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
1	0.0556	0.1250	0.0	0.0	0.0	0.0	2.2222	5.0	1.0000	68.222
2	0.0556	0.1250	0.0	0.0	0.0	0.0	2.2222	0.0	1.0000	18.222
3	0.0556	0.1250	0.0	0.0	0.0	0.0	0.0	0.0	1.0000	-4.0
4	0.0556	0.1250	0.0	0.0	0.0	0.0	-2.222	0.0	1.0000	18.222

Table 2.14 The Numerical Results of Figure C.2.

Cluster	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
1	0.0556	0.1250	0.0	0.0	0.0	0.0	2.2222	5.0	1.0000	68.222
2	0.0556	0.1250	0.0	0.0	0.0	0.0	2.2222	0.0	1.0000	18.222
3	0.0556	0.1250	0.0	0.0	0.0	0.0	0.0	0.0	1.0000	-4.0
4	0.0556	0.1250	0.0	0.0	0.0	0.0	-2.222	0.0	1.0000	18.222

Table 2.15 The Numerical Results of Figure C.3.

Cluster	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
1	0.0556	0.1250	0.0	0.0	0.0	0.0	2.2222	5.0	1.0000	68.222
2	0.0556	0.1250	0.0	0.0	0.0	0.0	2.2222	0.0	1.0000	18.222
3	0.0556	0.1250	0.0	0.0	0.0	0.0	0.0	0.0	1.0000	-4.0
4	0.0556	0.1250	0.0	0.0	0.0	0.0	-2.222	0.0	1.0000	18.222

Table 2.16 The Numerical Results of Figure C.6.

Cluster	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
1	0.0556	0.1250	0.0	0.0	0.0	0.0	2.2222	5.0	1.0000	68.222
2	0.0556	0.1250	0.0	0.0	0.0	0.0	2.2222	0.0	1.0000	18.222
3	0.0556	0.1250	0.0	0.0	0.0	0.0	0.0	0.0	1.0000	-4.0
4	0.0556	0.1250	0.0	0.0	0.0	0.0	-2.222	0.0	1.0000	18.222

Table 2.17 The Numerical Results of Figure C.7.

Cluster	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
1	0.0556	0.1250	0.0	0.0	0.0	0.0	2.2228	5.0008	1.0000	68.235
2	0.0556	0.1250	0.0	0.0	0.0	0.0	2.2222	0.0	1.0000	18.222
3	0.0556	0.1250	0.0	0.0	0.0	0.0	0.0	0.0	1.0000	-4.0
4	0.0556	0.1250	0.0	0.0	0.0	0.0	-2.223	0.0008	1.0000	18.234

Table 2.18 The Numerical Results of Figure C.8.

Cluster	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
1	0.0556	0.1250	0.0	0.0	0.0	0.0	2.2222	5.0	1.0000	68.222
2	0.0556	0.1250	0.0	0.0	0.0	0.0	2.2222	0.0	1.0000	18.224
3	0.0556	0.1250	0.0	0.0	0.0	0.0	0.0	0.0	1.0000	-4.0
4	0.0556	0.1250	0.0	0.0	0.0	0.0	-2.222	0.0	1.0000	18.222

Table 2.19 The Numerical Results of Figure C.9.

Cluster	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
1	0.0556	0.1250	0.0	0.0	0.0	0.0	2.2220	4.9997	1.0000	68.217
2	0.0556	0.1250	0.0	0.0	0.0	0.0	2.2225	0.0	1.0000	18.225
3	0.0556	0.1250	0.0	0.0	0.0	0.0	0.0003	0.0005	1.0000	-4.0
4	0.0556	0.1250	0.0	0.0	0.0	0.0	-2.222	0.0004	1.0000	18.225

Table 2.20 The CPU Time by Different Methods (unit: second).

Range Image	Figure C.1 (no noise)	Figure C.4 (plus 10% noise)	Figure C.5 (plus 20% noise)
RHT Method	354.6	698.0	1024.2
FRHT Method	230.2	258.3	298.5

Table 2.21 The Numerical Results of Figure C.10 by the RHT Method.

Cluster	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
1	0.0556	0.1250	0.0	0.0	0.0	0.0	2.2222	5.0	1.0000	68.222
2	0.0556	0.1250	0.0	0.0	0.0	0.0	2.2224	0.0	1.0000	18.225
3	0.0556	0.1250	0.0	0.0	0.0	0.0	0.0	0.0	1.0000	-4.0
4	0.0556	0.1250	0.0	0.0	0.0	0.0	-2.222	0.0	1.0000	18.222

Table 2.22 The Numerical Results of Figure C.10 by the FRHT Method.

Cluster	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
1	0.0556	0.1250	0.0	0.0	0.0	0.0	2.2222	5.0	1.0000	68.223
2	0.0556	0.1250	0.0	0.0	0.0	0.0	2.2222	0.0	1.0000	18.222
3	0.0556	0.1250	0.0	0.0	0.0	0.0	0.0	0.0	1.0000	-4.0
4	0.0556	0.1250	0.0	0.0	0.0	0.0	-2.222	0.0	1.0000	18.223

Table 2.23 The Numerical Results of Figure C.12.

Cluster	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
1	0.0573	0.1275	0.0003	-0.003	-0.001	-0.001	2.2442	5.0518	1.0000	68.925
2	0.0548	0.1234	0.0007	-0.001	0.0003	0.0015	2.1984	-0.015	1.0000	18.081
3	0.0548	0.1251	0.0	0.0011	0.0004	0.0016	0.0	0.0	1.0000	-3.994
4	0.0639	0.1357	0.0147	-0.003	0.0050	-0.003	-2.558	0.0669	1.0000	21.192

Table 2.24 Results for the Image in Figure C.1 by RHT and FRHT Methods with Approximate Distance.

Sampling Percentage	Number of Points	RHT			FRHT (3X3 windows)		
		*N*theo	*N*actual	CPU Time	*N*theo	*N*actual	CPU Time
100%	19600	564680	120747	194.4	8	8	13.2
80%	15667	564680	133748	212.1	8	9	12.9
60%	11792	564680	159604	241.1	8	8	12.5
50%	9875	564680	104126	164.3	8	11	12.4
40%	7953	564680	59876	100.6	8	10	12.0
20%	4025	564680	92233	143.5	8	9	11.7

CHAPTER 3

FUZZY CLUSTERING TECHNIQUES

3.1 Introduction

Fuzzy techniques have been successfully used for a variety of clustering and classification problems [Bezdek, 1981, Dave, 1989, Dave and Patel 1990a, 1990b]. They are based on objective function optimization techniques and generate a fuzzy partition of the data. The Fuzzy c-Means (FCM) algorithm [Bezdek, 1973] is one of the earliest fuzzy clustering methods proposed that is generally used to detect clusters of spherical shapes. It is the most widely used standard algorithm that might be changed to different variations by using different norm inducing matrices for distance measurement in it. The GK algorithm [Gustafson and Kessel, 1981] adopts a scheme where the norm inducing matrix is optimized. Bezdek et al [1981] generalized FCM by allowing the prototype to be linear manifolds of arbitrary and different dimensions. The Fuzzy c-Shells (FCS) algorithm [Dave et al, 1989, Dave, 1990] is a novel generalization of FCM that uses hyper-spherical shells as cluster prototypes. It has proved to be a great success as a method of detecting and representing circular sub-structure in two dimensional data-sets in digital images [Dave et al, 1990a, Dave, 1990]. Based on the "shell" concept, a series of algorithms have been developed, such as Fuzzy Ellipsoidal-Shell Clustering algorithm [Dave et al, 1990b], Adaptive Fuzzy C-Shells Clustering algorithm [Dave and Bhaswan, 1992, Dave and Fu,

1994], Fuzzy C Quadric Shells Clustering algorithm [Krishnapuram et al, 1992, Fu and Dave, 1993], etc..

Fuzzy clustering techniques partition a data point with membership values between *0* and *1*, and these values indicate the degrees to which a point belongs to different clusters. The clustering methods with hard memberships partition a point with only one of two values, either *0* or *1*. In other words, a point can only be classified to one cluster. Certainly, the hard clustering method can be taken as a special case of the fuzzy method. Based on the numerical experiments, the use of fuzzy memberships is shown to have a distinct advantage over hard memberships for those clustering algorithms [Dave, 1990]. The following discussion will focus on the fuzzy case except where indicated. In general, the fuzzy methods are found to be more reliable. The most of the proposed methods are used for 2-D image data. Here we present effective methods of 3-D object recognition from range images using fuzzy techniques.

3.2 Fuzzy C-Shells Clustering Algorithm

As earlier described, the prototype model and the definition of distance between a point and a prototype are two important factors for a clustering algorithm, so it is necessary to define them for the c-shells clustering methods before further discussion.

The spherical prototype and ellipsoidal prototype are two kinds of shell-like clusters. They can be characterized by the cluster centers. Since the spherical shell is a special case of ellipsoidal shell, where major radius and minor radius are equal, only a

general l-dimensional hyper-ellipsoidal shell is defined as below [Dave and Bhaswan, 1992].

Definition 1: The ith hyper-ellipsoidal shell prototype, p_i, at center q_i ($q_i \in \Re^l$) is the set

$$p_i = \{v \in \Re^l | (v - q_i)^T A_i (v - q_i) = 1\}, \tag{3.1}$$

where A_i is a $l \times l$ symmetric positive definite matrix that accounts for the size, eccentricity and orientation of the ellipsoid. When A_i is a diagonal matrix with all diagonal elements being equal, the definition is applicable to hyper-spherical shells.

The definition of distance between a point and a shell cluster is given below.

Definition 2: The distance D_{ik} between a point v_k and a hyper-ellipsoidal shell p_i is defined as

$$(D_{ik})^2 = ([(v_k - q_i)^T A_i (v_k - q_i)]^{1/2} - 1)^2 \tag{3.2}$$

The definition is also applicable to hyper-spherical shells when A_i is a diagonal matrix with equal entries on diagonal.

As stated in chapter 1, an objective function can be formulated based on the defined prototype and the distances. Assume that $V = \{v_1, v_2, \ldots, v_n\} \subset \Re^l$ is a set of n feature vectors in the feature space, $Q = \{q_1, q_2, \ldots, q_c\} \subset \Re^l$ represents a set of shell centers, and $A = \{A_1, A_2, \ldots, A_c\}$ indicates a set of $l \times l$ symmetric positive definite

matrices. Let u_{ik} be the grade of membership of feature point v_k in cluster p_i, and $U=[u_{ik}]$ is a $c \times n$ matrix called the fuzzy partition matrix satisfying the following conditions:

$$u_{ik} \in [0,1], \quad i \in \{1,2,...,c\}, k \in \{1,2,...,n\},$$

$$\sum_{i=1}^{c} u_{ik} = 1, \quad k \in \{1,2,...n\},$$

$$0 < \sum_{k=1}^{n} u_{ik} < n, \quad i \in \{1,2,...c\}, \tag{3.3}$$

For hard partitions, the only difference from above conditions is $u_{ik} \in \{0,1\}$.

Then the problem is to find c clusters as well as a c-partition of the n feature vectors by minimizing the following objective function:

$$J(U,Q,A) = \sum_{i=1}^{c}\sum_{k=1}^{n}(u_{ik})^m (D_{ik})^2, \tag{3.4}$$

where, $m \in [1,\infty)$ is a weighting exponent called the fuzzifier. Minimization of the objective function subject to the necessary constraints yields the following equations.

For fuzzy memberships:

$$u_{ik} = \begin{cases} 1/dd & I_k = \varnothing \\ 0 \quad i \notin I_k, & I_k \neq \varnothing \\ 1 \quad i \in I_k, & I_k \neq \varnothing \end{cases} \tag{3.5}$$

where $dd = \sum_{j=1}^{c} (\frac{D_{ik}}{D_{jk}})^{\frac{2}{m-1}}$, $I_k = \{i | 1 \le i \le c, D_{ik}{}^2 = 0\}$.

For hard memberships:

$$u_{ik} = \begin{cases} 0 & i \notin I_k \\ 1 & i \in I_k \end{cases} \tag{3.6}$$

where $I_k = \{i | D_{ik} = \min(D_{lk}, 1 \le l \le c)\}$. And the following equations about cluster centers q_i and norm inducing matrices A_i are satisfied for either case of memberships:

$$\sum_{k=1}^{n} (u_{ik})^m \frac{D_{ik}}{d_{ik}} (v_k - q_i) = 0, \tag{3.7}$$

$$\sum_{k=1}^{n} (u_{ik})^m \frac{D_{ik}}{d_{ik}} (v_k - q_i)(v_k - q_i)^T = 0, \tag{3.8}$$

where d_{ik} is defined as

$$(d_{ik})^2 = (v_k - q_i)^T A_i (v_k - q_i). \tag{3.9}$$

Therefore, the solution to q_i and A_i can be obtained by simultaneously solving equations (3.7) and (3.8). For hyper-spherical shells, equation (3.9) becomes a scalar and provides radius information. Now, the fuzzy c-shells (FCS) algorithm can be summarized as below.

FCS Algorithm:

(1) Fix the number of clusters c and the exponent m, $m \in [1,\infty)$ and $m = 1$ for hard memberships. Initialize the fuzzy partition matrix U.

(2) Compute q_i and A_i by equations (3.7), (3.8) and other related equations.

(3) Update the fuzzy membership matrix U according to equation (3.5) or (3.6). If the fuzzy partition is stable, stop, otherwise, go to step (2).

3.3 Fuzzy C Quadric Shells Clustering Algorithm

Based on fuzzy c-shells clustering methods, the shell is extended to the quadric type and the corresponding c quadric shells clustering algorithm has been used in 2-D image data

[Krishnapuram et al., 1992, Dave and Bhaswan, 1992]. The following will discuss how to apply the Fuzzy C Quadric Shells clustering method to 3-D range data.

Generally, a quadric curved surface in 3-D space can be represented as

$$a_1x^2 + a_2y^2 + a_3z^2 + a_4xy + a_5xz + a_6yz + a_7x + a_8y + a_9z + a_{10} = 0. \qquad (3.10)$$

Obviously, there are ten parameters for each surface, and nine of them are independent. The hyper-spheres and hyper-ellipsoids are extreme cases of it. The definition for the quadric prototype in 3-D space is given as below.

Definition 3: The *i*th quadric curved surface prototype in 3-D space is the set

$$p_i = \{v \in \Re^3 | v^T A_i v + v^T B_i + C_i = 0\}, \qquad (3.11)$$

where

$$v^T = [x, y, z], \qquad (3.12)$$

and

$$A_i = \begin{bmatrix} a_{i1} & a_{i1}/\sqrt{2} & a_{i5}/\sqrt{2} \\ a_{i1}/\sqrt{2} & a_{i2} & a_{i6}/\sqrt{2} \\ a_{i5}/\sqrt{2} & a_{i6}/\sqrt{2} & a_{i3} \end{bmatrix}, B_i = \begin{bmatrix} a_{i7} \\ a_{i8} \\ a_{i9} \end{bmatrix}, C_i = a_{i10}. \qquad (3.13)$$

Equation (3.10) is different from the above defined prototype, where the constant $\sqrt{2}$ is introduced for simplifying the following derivation. Correspondingly, the distance from a point to a cluster can be defined as below.

Definition 4: The distance d_{ik} between a point v_k and a cluster p_i is defined as

$$(d_{ik})^2 = (v_k^T A_i v_k + v_k^T B_i + C_i)^2. \qquad (3.14)$$

The d_{ik} is an algebraic distance instead of an exact Euclidean distance.

Then, with the same assumptions as in the FCS method for fuzzifier m and fuzzy partition matrix U, the objective function to be minimized is

$$J_Q = \sum_{i=1}^{c}\sum_{k=1}^{n}(u_{ik})^m (d_{ik})^2 . \tag{3.15}$$

Rewrite it as

$$d_{ik}^2 = \Gamma_i^T M_k \Gamma_i, \tag{3.16}$$

where

$$\Gamma_i^T = \left[\Gamma_{i1}^T | \Gamma_{i2}^T\right], \tag{3.17}$$

and

$$\Gamma_{i1}^T = \begin{bmatrix} a_{i1} & a_{i2} & a_{i3} & a_{i4} & a_{i5} & a_{i6} \end{bmatrix}, \tag{3.18}$$

$$\Gamma_{i2}^T = \begin{bmatrix} a_{i7} & a_{i8} & a_{i9} & a_{i10} \end{bmatrix}. \tag{3.19}$$

In equation (3.16),

$$M_k = \begin{bmatrix} O_k & R_j^T \\ R_k & S_k \end{bmatrix}, \tag{3.20}$$

where

$$O_k = o_k o_k^T, \quad S_k = s_k s_k^T, \quad R_k = s_k o_k^T, \tag{3.21}$$

and

$$o_k^T = \begin{bmatrix} x_k^2 & y_k^2 & z_k^2 & \sqrt{2}x_k y_k & \sqrt{2}x_k z_k & \sqrt{2}y_k z_k \end{bmatrix},$$

$$s_k^T = \begin{bmatrix} x_k & y_k & z_k & 1 \end{bmatrix}. \tag{3.22}$$

Now, the objective function becomes

$$J_Q = \sum_{i=1}^{c}\sum_{k=1}^{n}(u_{ik})^m \Gamma_i^T M_k \Gamma_i. \tag{3.23}$$

In order to avoid the trivial solution, we can use the following constraint [Krishnapuram et al., 1992]:

$$\|\Gamma_{i1}\|^2 = 1. \tag{3.24}$$

Let

$$F_i = \sum_{j=1}^{n}(u_{ij})^m O_j, \quad G_i = \sum_{j=1}^{n}(u_{ij})^m R_j, \quad H_i = \sum_{j=1}^{n}(u_{ij})^m S_j, \tag{3.25}$$

thereafter, the Γ_{i1} and Γ_{i2} can be obtained from the matrix $(F_i - G_i^T H_i^{-1} G_i)$. Let $\Xi_{\min}$ be the eigenvector of the matrix associated with the smallest eigenvalue, then

$$\Gamma_{i1} = \Xi_{\min}, \tag{3.26}$$

$$\Gamma_{i2} = -H_i^{-1} G_i \Gamma_{i1}. \tag{3.27}$$

The memberships will still be updated using equation (3.5) for the fuzzy case or (3.6) for the hard case, except that the distances are computed using equation (3.14). This algorithm is summarized as:

FCQS Algorithm:

(1) Fix the number of clusters c and the exponent m, $m \in [1,\infty)$ and $m = 1$ for hard memberships. Initialize the fuzzy partition matrix U.

(2) Compute the parameter vector Γ_i according to equations (3.26), (3.27) and other related equations.

(3) Update the fuzzy partition matrix U by equation (3.5) or (3.6). If the cluster partition is stable, stop, otherwise, go to step (2).

3.4 The Improved Fuzzy C-Shells Clustering Algorithms

In the above FCQS algorithm, for quadric surfaces, the distance defined by Equation (3.14) is a kind of nonlinear algebraic distance instead of Euclidean distance. As an example in Figure 3.1, there are four points $A(0,6)$, $B(0,4)$, $C(2,0)$, $D(6,0)$ and two ellipses E_1: $x^2/16+y^2/4=1$, E_2: $(x-10)^2/4+y^2=1$. The Euclidean distances D_E and the algebraic distances D_A from these points to the two ellipses are respectively shown in Table 3.1.

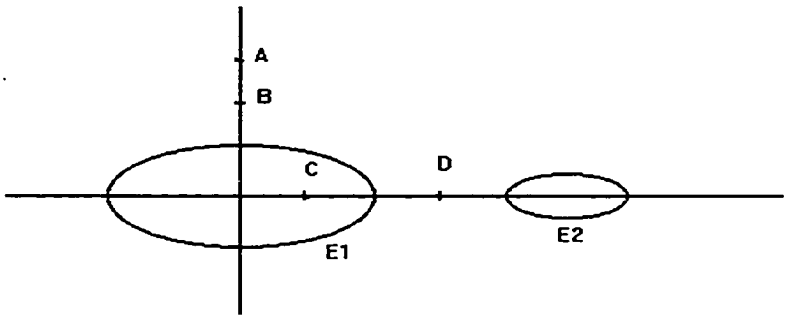


Figure 3.1 Two ellipses for the distance comparison

From Table 3.1, we have the following findings:

- The algebraic distance is not proportional to the Euclidean distance;
- The algebraic distance is sensitive to the location of point, such as the distances from B, C, D to ellipse E_1;

- The algebraic distance is different for different curves, such as the distances from point D to two ellipses.

Therefore, the memberships updated by the nonlinear distance are not very meaningful. Moreover, the nonlinear distances make the curve fitting rather sensitive to other curves in the data set, which might lead to unstable solutions [Krishnapuram et al, 1993].

Table 3.1 The Comparison Among Euclidean Distance, Algebraic Distance, and Approximate Distance for Figure 3.1.

Ellipse	E_1				E_2	
Point	A	B	C	D	C	D
D_E	4	2	2	2	6	2
D_A	8	3	0.75	1.25	15	3
D_P	2.67	1.5	3.0	1.67	3.75	1.5

In general, the distance defined by Equation (3.14) is nonlinear. But as a special case when the prototypes are planes, the model becomes a first-order polynomial function

$$Ax + By + Cz + D = 0, \tag{3.28}$$

then the Euclidean distance from a point (x_0, y_0, z_0) to the plane prototype is

$$(D_E)^2 = \frac{(Ax_0 + By_0 + Cz_0 + D)^2}{A^2 + B^2 + C^2}, \tag{3.29}$$

and the algebraic distance is

$$(D_A)^2 = (Ax_0 + By_0 + Cz_0 + D)^2. \tag{3.30}$$

Obviously, the algebraic distance becomes linear.

For the prototype with highly nonlinear algebraic distance, the FCQS method might not work well and a new algorithm using Euclidean distance might improve the clustering performance. Equation (2.16) gives the formula for the computation of

Euclidean distance between a point and a quadric prototype in 3D space. It is necessary to point out that for simplicity, the Euclidean distance is only used to update the memberships and the parameters are still computed using the algebraic distance [Krishnapuram et al., 1992]. The improved FCQS algorithm with exact distances can be described as:

The Improved FCQS Algorithm with Exact Distances:

(1) Fix the number of clusters c and the exponent m, $m \in [1,\infty)$ and $m = 1$ for hard memberships. Initialize the fuzzy partition matrix U.

(2) Compute the parameter vector Γ_i according to equations (3.26), (3.27) and other related equations.

(3) Update the fuzzy partition matrix U by equation (3.5) or (3.6), where the distance is computed using Equation (2.16). If the cluster partition is stable, stop, otherwise, go to step (2).

Because the computation of Euclidean distance is very time consuming, it is suggested to use the regular FCQS method whenever possible. Fortunately, some approaches using approximate distance to replace exact distance have been performed [Taubin 1991, Haralick and Shapiro 1992]. What interests us is the first-order approximation of the exact distance by which the approximate distance of a point from a surface can be represented as:

$$(d_{Aik})^2 = \frac{(d_{ik})^2}{|\nabla d_{ik}|^2} \tag{3.31}$$

where ∇d_{ik} is the gradient of the algebraic distance functional d_{ik} evaluated at point k. Undoubtedly, the approximate distance is still a kind of nonlinear distance. For the example shown in Figure 3.1, the approximate distances D_P between the four points and the two ellipses are computed according to Equation (3.31) and listed in Table 3.1, which clearly indicates the nonlinearity of approximate distance. Nevertheless, it can work well in various applications and its performance will be further investigated in Chapter 5.

Equation (3.31) brings a closed form of distance solution to our algorithms so the computation speed might be extremely raised, and correspondingly, the improved FCQS algorithm with approximate distances can be described as:

The Improved FCQS Algorithm with Approximate Distances:

(1) Fix the number of clusters c and the exponent m, $m \in [1,\infty)$ and $m = 1$ for hard memberships. Initialize the fuzzy partition matrix U.

(2) Compute the parameter vector Γ_i according to equations (3.26), (3.27) and other related equations.

(3) Update the fuzzy partition matrix U by equation (3.5) or (3.6), where the distance is computed using Equation (3.31). If the cluster partition is stable, stop, otherwise, go to step (2).

3.5 Experiments and Conclusions

The FCS algorithm has been widely used in various applications [Dave 1990, Dave and Patel 1990a, 1990b, Dave and Bhaswan 1992, etc.]. In what follows, some experiments for the FCQS method are presented.

The range image in Figure C.1 contains four clusters with 19600 data points. Using ideal initial memberships, i.e., the memberships of a point are assigned as 1 to the right cluster and 0 to other clusters, the FCQS algorithm converges in one iteration when the fuzzifier m is 2. While the largest membership of a point to its right cluster is assigned as 0.7, the cluster partition becomes stable in 7 iterations when $m = 5$ and in 3 iterations when $m = 10$, but does not work well when $m = 2$. Furthermore, when the largest membership of a point to its right cluster is assigned as 0.5, the cluster partition becomes stable in 6 iterations when $m = 10$, but the algorithm does not work well when $m = 2$ and $m = 5$. To see the performance of the improved FCQS algorithm with exact distances, when the largest membership of a point to its right cluster is assigned as 0.7, the cluster partition becomes stable in 7 iterations when $m = 5$ and in 4 iterations when $m = 10$, but algorithm does not work well when $m = 2$. Based on the experiments, the CPU time for one iteration is usually about 4 to 5 times of the time spent by the normal FCQS algorithm. However, it is necessary to point out that the method of Euclidean distance computation used here is based on the Lagrange multiplier technique and the routines from the MINPACK (see Appendix A for more details), the speed might be increased by other methods.

For the image in Figure C.4 with 10% extra random noise points, while the largest membership of a point to its right cluster is assigned as 0.7, the FCQS algorithm may not be able to work well when m is a smaller number. When m is as large as 25, the FCQS method becomes convergent in 12 iterations and each iteration uses about 150 seconds on a SPARC 10 computer, whereas the improved FCQS algorithm with exact distances becomes stable in 17 iterations. For one iteration, the CPU time by the improved FCQS algorithm with exact distances is about 4 to 5 times of the time by the normal FCQS algorithm.

Similarly, we also tested the FCQS algorithm with approximate distances and found that it had the same level of computation speed as the FCQS algorithm with algebraic distances.

Through the experiments, we can see that the FCQS algorithm uses all the data information to compute the parameters for all clusters, the results can be very precise when the initial memberships are accurate enough. Its computation speed for an iteration is proportional to the number of points and the number of clusters, and the number of iterations mostly depends on the initial memberships and the characteristics of each cluster model. With different distance computation methods, the number of iterations needed is different but the FCQS with exact distances is more robust than other two methods. The major limitations of such kind of algorithms are that the number of clusters and a better initial membership matrix need to be given in advance, and their performance in noise images is poor. The later chapters will further investigate these issues.

CHAPTER 4

NOISE PROBLEM WITH CLUSTERING TECHNIQUES

4.1 Introduction

Clustering techniques are used to identify components or data points belonging to different assemblies or different clusters. Many investigators have developed various clustering methods for different applications [Everitt, 1974, Jain and Dubes, 1988]. Since clustering techniques deal with numerical problems, the related mathematical theory has been developed based on the various demands from practical applications and the different clustering methods have also emerged as an outgrowth of developments in related theory. As we know, noisy data is always mixed with original data, thus any potential method must be able to deal with noisy data.

To deal with noise problem, there are two kinds of methods. One is to identify the noisy data points and remove them before applying any clustering methods [Jain and Dubes, 1988]. Here, the clustering methods do not have to deal with noise problem, but identifying noisy data may be extremely difficult. Weiss [1988] uses the maximum likelihood principle to separate the data of interest from random noise and then fits a single line to good data amongst noisy background. The technique has a disadvantage of becoming trapped at local minima. The other kind of methods are based on the clustering algorithms designed to deal with the noisy data points, namely, the algorithms by themselves can detect good clusters amongst noisy data. As an example, Jolion and

Rosenfeld [1989] detect good clusters amongst noisy points by using a weighting method which assigns higher weights to the points belonging to the good clusters and lower weights to the noisy points, where the weight is proportional to the density of data points in its vicinity.

In chapter 3, Equation (3.3) shows us that for any point in the data, the sum of memberships for all clusters must be one even when that point is a noisy point. It would be more reasonable if the sum of memberships of a noisy point in all good clusters is less than one or even is zero. To solve this problem, Dave [1991] proposed a concept of 'noise-prototype' that is presented as below.

Definition: Noise prototype is a universal entity such that it is always at the same distance from every point in the data-set. Let p_n be the noise prototype and v_k be the point in feature space. Then the noise prototype is such that d_{nk}, the distance of point v_k from p_n, is

$$d_{nk} = \psi, \quad \forall k. \tag{4.1}$$

This definition implies that all the points in the data-set are at the same distance from the noise cluster though it does not indicate what the distance ψ is. Applying this concept to the previously described fuzzy algorithms can create new noise fuzzy clustering methods that will be more effective for handling noisy data and outliers. The following sections will present Noise Fuzzy C-Shells (NFCS) clustering method and Noise Fuzzy C Quadric Shells (NFCQS) clustering method.

4.2 Noise Fuzzy C-Shells Clustering Algorithm

Based on the concept of 'noise prototype,' the FCS algorithm can be re-formulated by adding a noise cluster. Assume that there are c good clusters in the data-set, the extra $(c+1)$th cluster is a noise cluster. Then the objective function including the noise cluster is rewritten as:

$$J_n(U,Q,A)=\sum_{i=1}^{c+1}\sum_{k=1}^{n}(u_{ik})^m(D_{ik})^2, \tag{4.2}$$

where the distances are defined as:

$$(D_{ik})^2=\begin{cases}([(v_k-q_i)^T A_i(v_k-q_i)]^{1/2}-1)^2 & k\in\{1,2,\ldots,n\}, i\in\{1,2,\ldots,c\}\\ \psi^2 & i\in\{c+1\}\end{cases} \tag{4.3}$$

As to the fuzzy partition matrix, the constraint on the memberships becomes

$$0\le\sum_{i=1}^{c}u_{ik}\le 1, \qquad k\in\{1,2,\ldots n\} \tag{4.4}$$

This is the major difference between the FCS method and the NFCS method. With the constraint (4.4), a noisy point may be assigned a total membership close to zero in all good clusters. This obviously makes more sense.

Assume that the noise distance ψ is specified, the minimization of J_n brings up the following equations.

For fuzzy memberships:

$$u_{ik}=\begin{cases}1/dd & I_k=\varnothing\\ 0 \quad i\notin I_k, & I_k\neq\varnothing\\ 1 \quad i\in I_k, & I_k\neq\varnothing\end{cases} \tag{4.5}$$

where $dd=\sum_{j=1}^{c+1}(\frac{D_{ik}}{D_{jk}})^{\frac{2}{m-1}}$, $I_k=\{i|1\le i\le c+1, D_{ik}{}^2=0\}$.

For hard memberships:

$$u_{ik} = \begin{cases} 0 & i \notin I_k \\ 1 & i \in I_k \end{cases}, \tag{4.6}$$

where $I_k = \{i | D_{ik} = \min(D_{lk}, 1 \le l \le c+1)\}$. And the following equations about cluster centers q_i and norm inducing matrices A_i are satisfied for either case of memberships:

$$\sum_{k=1}^{n} (u_{ik})^m \frac{D_{ik}}{d_{ik}} (v_k - q_i) = 0, \tag{4.7}$$

$$\sum_{k=1}^{n} (u_{ik})^m \frac{D_{ik}}{d_{ik}} (v_k - q_i)(v_k - q_i)^T = 0, \tag{4.8}$$

where $i \in \{1,2,\ldots,c\}$. Similarly as the FCS method, the solution to q_i and A_i can be obtained by simultaneously solving equations (4.7) and (4.8).

In the NFCS algorithm, the noise distance ψ is a critical parameter, which can be estimated as [Dave, 1991]:

$$\psi^2 = \lambda * \left[\frac{\sum_{i=1}^{c} \sum_{k=1}^{n} (D_{ik})^2}{nc} \right] \tag{4.9}$$

where λ is a multiplier used to obtain ψ from the average of distances.

Now, the Noise Fuzzy C-Shells (NFCS) algorithm can be summarized as below.

NFCS Algorithm:

(1) Fix the number of clusters c and the exponent m, $m \in [1,\infty)$ and $m = 1$ for hard memberships. Set the initial λ and initialize the fuzzy partition matrix U.

(2) Compute q_i and A_i by equations (4.7), (4.8) and other related equations.

(3) Calculate the distance using equation (4.3) and update the fuzzy membership matrix U according to equation (4.5) or (4.6). If the fuzzy partition is stable, stop, otherwise, go to step (2).

4.3 Noise Fuzzy C Quadric Shells Clustering Algorithm

Applying the concept of 'noise prototype' to the FCQS algorithm, a new algorithm called Noise Fuzzy C Quadric Shells (NFCQS) clustering method is created. Let the $(c+1)$th cluster be the noise cluster, the objective function is then rewritten as:

$$J_{NQ} = \sum_{i=1}^{c+1}\sum_{k=1}^{n}(u_{ik})^m(d_{ik})^2 \tag{4.10}$$

where the distance is defined as:

$$(d_{ik})^2 = \begin{cases} \Gamma_i^T M_k \Gamma_i & k \in \{1,2,\ldots,n\}, i \in \{1,2,\ldots,c\} \\ \psi^2 & i \in \{c+1\} \end{cases} \tag{4.11}$$

The equation (4.9) about noise distance is still suitable here. Using the same definitions as in the FCQS algorithm for parameter vector Γ_i and matrix M_k, then the objective function becomes

$$J_{NQ} = \sum_{i=1}^{c+1}\sum_{k=1}^{n}(u_{ik})^m \Gamma_i^T M_k \Gamma_i \tag{4.12}$$

Same as the FCQS method, the parameter vector Γ_i can be computed by equations (3.26) and (3.27). As to the memberships, the constraints and updating equations for the NFCS method are applicable to the NFCQS algorithm here. The algorithm is described as below.

NFCQS Algorithm:

(1) Fix the number of clusters c and the exponent m, $m \in [1,\infty)$ and $m = 1$ for hard memberships. Set the initial λ and initialize the fuzzy partition matrix U.

(2) Compute the parameter vector Γ_i according to equations (3.26), (3.27) and other related equations.

(3) Calculate the distance using equation (4.11) and update the fuzzy partition matrix U by equation (3.5) or (3.6). If the cluster partition is stable, stop, otherwise, go to step (2).

4.4 Experiment Results and Conclusions

To evaluate the performance of the NFCS algorithm, one example is presented here to detect the cluster prototypes in noise data. Figure D.1 shows three circular clusters that are generated using random distribution of points centered around the prototypes listed in Table 4.1 plus some randomly added noise. The number of noisy points is about the same as the number of good points. The prototypes detected using the conventional FCS algorithm are shown in Figure D.2, and the prototypes detected using the NFCS technique are shown in Figure D.3. To be fair in comparison, very good initial estimates of the circle position and size were provided for both the algorithms. Results are even worse for the conventional FCS algorithm if random initialization is used. The numerical values of the detected prototypes are shown in Table 4.1. This example clearly shows that the NFCS method is robust against noise.

As to the NFCQS algorithm, an example of the range image containing four clusters and 10% extra noise points in Figure C.4, which is used in Chapter 3, is presented here. As stated in Chapter 3, for the noise image, the detection results using the FCQS method are stable in 12 iterations when $m = 25$. While applying the NFCQS algorithm to the image with the same m value, the results become stable in 9 iterations. Therefore, the noise type algorithms might even increase converging speed.

Table 4.1 Numerical Results for the Example in Figure D.1.

Prototype No.	Actual Values		Computed Values: FCS		Computed Values: NFCS	
	Center (x,y)	Radius	Center (x,y)	Radius	Center (x,y)	Radius
1	(41.7,158.4)	40.08	(68.9,136.8)	56.13	(43.8,157.3)	40.74
2	(145.9,184.5)	49.77	(145.9,184.1)	49.68	(145.8,184.0)	49.61
3	(186.6,109.6)	39.93	(188.7,99.0)	43.44	(187.1,106.3)	40.75

As a summary, the clustering algorithms combining the fuzzy c shell techniques and the concept of noise prototype possess better performance in images with noise, and might even result in faster convergence. Nevertheless, the number of clusters is still required to be known in advance, and better initialization of the partition matrix is required in some cases. Chapter 5 will present a novel algorithm to overcome these disadvantages.

CHAPTER 5

INTEGRATED CLUSTERING METHOD

In the early chapters, the Hough transform based clustering methods as well as the fuzzy c-shells and the noise fuzzy c-shells clustering algorithms are investigated. Each kind of method has its advantages and disadvantages. Based on further analysis of these methods, in this chapter, we develop an integrated method such that their advantages could be combined and utilized, and their disadvantages might be avoided.

5.1 Analysis of Different Clustering Algorithms

Comparing with other Hough type clustering methods, the FRHT possesses the advantages of high detection speed, low storage requirement, and arbitrarily parameter precision. Our experience with the FRHT method indicates that the false iterations and the higher minimum number of points for a cluster are two main reasons that contribute to increase the computation load, especially for the cases of images with noise points. Here the false iteration means an iteration by which a right cluster can not be found. No matter if it is a true or a false iteration, the FRHT would calculate the distances between all the remaining points in the data set and a possible cluster. Since the exact distance computation is very time consuming, a false iteration would increase the computations. The higher minimum number of points for a cluster could make the detected clusters more reliable but might also increase the number of false iterations, whereas the lower minimum number could increase the detection speed but might create false clusters. So there is a trade off between the computation load and the precision of detection results.

Furthermore, sometimes even with a higher threshold minimum number of points, the FRHT might not be able to detect all clusters.

For the fuzzy c-shells clustering algorithms and noise type algorithms, the computing time mainly depends on the number of distance computations that is proportional to the number of iterations, the number of clusters and the total number of feature points. Assume that N_{iter} iterations is needed to obtain a stable membership matrix, then the total number of distance computations by the fuzzy c-shells clustering algorithms is

$$N_{iter} \times num \times noc, \tag{5.1}$$

and the noise type algorithms generally need a few more computation than that. However, as indicated in Chapter 4, the noise type algorithms might need fewer iterations. Therefore in practice, they have the same level of computation load. The major difference between them is that the noise type methods might be able to treat noise points more properly. Through experiments, we found that the number of iterations was roughly proportional to the number of clusters for the fuzzy clustering methods. Therefore, the total number of distance computations is roughly proportional to the square of the number of clusters. It is not difficult to see that the number of distance computations will reduce quickly with the number of clusters reducing. However, as indicated in earlier chapters, for the fuzzy c-shells clustering algorithms and their noise type methods, the number of clusters and the good initial memberships have to be given in advance. As for the FRHT method, when an appropriate number of windows is selected, the number of random mappings is very small and the detection time mainly relies on the total number of distance computations that is

proportional to the number of data points and the number of clusters. Assume that there are not any false iterations and each cluster contains the same number of points, then the total number of distance computations can be represented as

$$num \times \frac{noc+1}{2}. \tag{5.2}$$

Equations (5.1) and (5.2) show us that the FRHT method generally has higher detection speed than the fuzzy clustering methods when they use same distance computation method. Unfortunately, the FRHT algorithm could not work well when the algebraic distance is used. Moreover, when the data set contains noise data points, the computing load and the false iterations will increase, and the FRHT method itself may not work well.

The above arguments can be summarized as follows: the FRHT method does not need the number of clusters and the initial memberships in advance, and possesses higher computation speed; the noise fuzzy c-shells algorithms can work fast and accurately with the noise data, especially when there is only one cluster. These features give us an opportunity to combine them into a novel clustering algorithm that could omit the initial conditions and achieve accurate and fast results. In the FRHT method, after the score value at a point in the accumulating binary tree reaches the threshold n_t, it starts checking all the points whether they belong to the possible cluster. If the number of points belonging to a possible cluster is more than the predefined minimum number of points for a cluster, the possible cluster becomes a true cluster, otherwise, the possible cluster is a false one. In reality, due to the noise effect, a true cluster might be omitted because of a higher predefined minimum number of points for a cluster, and a false cluster might be obtained because of a lower predefined number. If the noise fuzzy c-shells clustering

algorithm is applied here to refine and ascertain the results, it would reduce the false iterations and improve the reliability of results. Under such conditions, the problem for the noise fuzzy c-shells clustering methods becomes a simple one in which the data set contains only one cluster with good initial parameters. The next section will present the new algorithm in detail.

5.2 The Integrated Clustering Method

Upon the above discussion, it can be concluded that the novel integrated algorithm would greatly improve the performance of cluster detection. Figure 5.1 is the flow diagram of this algorithm and the details are described as following.

The only difference between Figure 2.1 and Figure 5.1 is that the later one has an extra block for refining results by applying the noise fuzzy clustering methods. To avoid the repeat, the description about most of the integrated algorithm is referred to the FRHT method in section 2.3 except what is presented below.

After a possible set of parameters is obtained, the threshold distance D_t for the integrated method can be set as a greater value than that in the FRHT. With the greater threshold distance D_t, more points could be collected to compose a new data set that would be used to refine the results by the NFCQS algorithm. Usually, the collected points in the new data set should be more than the minimum number of points for a cluster, otherwise, they should be returned to the original data set and another searching iteration begins. Because the new data set contains only one possible cluster and the parameters for the possible cluster are obtained by the previous step, the number of clusters for the

NFCQS is one and the initial memberships can be calculated using the previously computed parameters. Therefore, the NFCQS algorithm is reformed for the new data set as below.

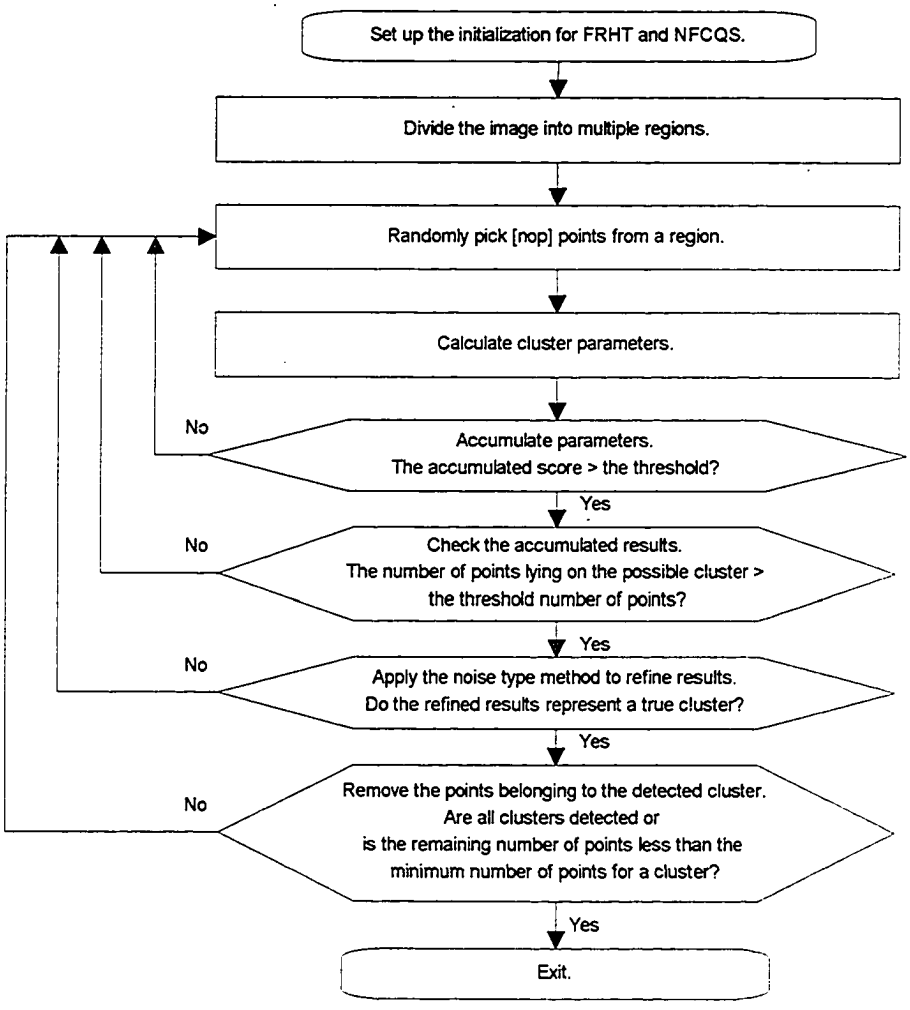


Figure 5.1 The Integrated Clustering Algorithm.

NFCQS Algorithm for One Cluster:

(1) Fix the number of clusters $c = 1$ and the exponent m, $m \in [1,\infty)$ and $m = 1$ for hard memberships. Set the initial λ .

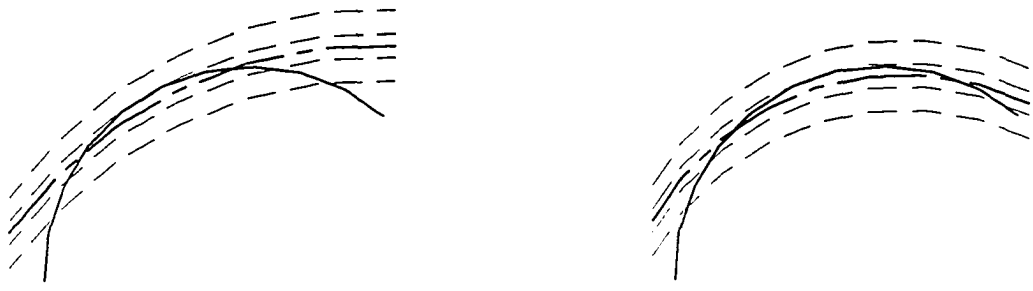
(2) Initialize the fuzzy partition matrix U by equation (3.5) or (3.6).

(3) Compute the parameter vector Γ_i according to equations (3.26), (3.27) and other related equations.

(4) Calculate the distance using equation (4.11) and update the fuzzy partition matrix U by equation (3.5) or (3.6). If the cluster partition is stable, stop, otherwise, go to step (3).

As soon as the parameters are refined by the NFCQS method, they can be used to check how many points belong to the possible cluster among all the unclassified points, where the same threshold distance D_t and the same minimum number of points for a cluster as in the FRHT method are used. Sometimes, as the initial cluster may only include partial true cluster as shown in Figure 5.2a, the refined results by the NFCQS may not include the complete true cluster yet, just as shown in Figure 5.2b. In this case, it is necessary to perform another NFCQS operation based on the last NFCQS results, which could be continued until the true cluster is found. Of course, to determine whether a true cluster is found in the algorithm, we can check the difference of number of points in the cluster between two consecutive NFCQS operations, and if the difference is negligible, the further operations are not necessary and the true clusters is found, otherwise, the further operations are needed. Normally, the number of points in a cluster increases with every

further NFCQS operations, but it might even decrease in some situations. If the decrease is negligible, the true parameters are found and the process stops, otherwise, the search process should be discarded and a new FRHT searching process should be started.



(a) (b)

Solid line: true cluster, center line: estimated cluster, dashed line: distance boundary.

Figure 5.2 The NFCQS Refines the Results. The two inner dashed lines are +/-D_t boundaries around the estimated cluster, and the two outer dashed lines are the greater distance boundaries so more points could be collected for NFCQS. (a) Initial NFCQS process. (b) Further NFCQS process.

In Chapter 3, three different distance computation methods are discussed. While using algebraic distances and approximate distances, the NFCQS has same level of computation speed that is much faster than the speed when the exact distances are used. Experiments show that the NFCQS using algebraic distances works very well in many applications, especially when a better initial membership matrix is available and there is only one cluster in the data set. Therefore, while applying the NFCQS to refine the results by the FRHT method, we normally use algebraic distances by which detection would be faster.

When exact distances are used in the FRHT method, the nonlinear problem with distance computation is avoided but the computation speed is too slow and is not practical. Though the approximate distances are somewhat nonlinear as indicated in Chapter 3, they can work well in some circumstances, especially when the points are quite close to the boundary. Some efforts have been conducted on this issue [Taubin 1991, Keren et al. 1994]. Hence, we usually use the approximate distances in the FRHT.

It seems that the new integrated method increases the computation load as it calculates the distances at least twice, which are right before and after applying the NFCQS algorithm respectively. On the other hand, the integrated method greatly reduces the false iterations through use of the NFCQS algorithm, especially for the noisy data sets. Moreover, the new method is a more reliable clustering technique that can achieve more accurate results and detect the clusters omitted by other methods. Its performance is presented in the following section.

5.3 Experiments and Results

In what is presented below, the algebraic distances are used for the NFCQS iterations execpt where indicated, and as for the FRHT method and the FRHT process in the integrated method, we use both exact distances and approximate distances.

While using the FRHT method, we found that sometimes it was sensitive to the choice of the number of regions, and sometimes it did not work well for the noise images. Figure E.1 is the detection result of the image in Figure C.5 by the FRHT method using 3×3 regions. Each cluster is displayed with a different gray value. With a SPARC 10

Station, the CPU time spent for the detection is 339.7 seconds while the exact distances are used. There are five clusters in Figure E.1, which are listed in Table 5.1. From the table, we can see the parameters of clusters 4 and 5 are very close as they actually belong to one cluster. Applying the integrated algorithm to the same image with the same initial conditions, we can obtain the graph results as shown in Figure E.2 and the parameter results as listed in Table 5.2. There are four clusters which are in a close match with the original image. However, due to the extra Euclidean distance computations performed for the huge amount of points, the CPU time spent by the integrated algorithm is 570.4 seconds, which is more than the time used by the FRHT method.

As stated in Chapter 2, the image in Figure C.5 is generated by randomly adding some noise points to a non-noise image. The simplified noise image in Figure C.11 is generated by shifting the original points within a limited distance. For this image, the FRHT method works but with a slow computation speed, as discussed in Chapter 2. Applying the integrated clustering algorithm to it with the same initial conditions, we obtain the graph results as shown in Figure E.3 and the parameter results as listed in Table 5.3. The CPU time used for the detection is 49.9 seconds that is less than 146.2 seconds, the time spent by the normal FRHT method, where the exact distances are used for both FRHT processes. Therefore, the integrated clustering algorithm can improve the clustering results and sometimes can even increase the computation speed.

What is presented above are the examples from the artificial generated data images. The following will show the clustering ability of the integrated algorithm for real, practical range images. Figure E.4 shows a range image generated by adding some noise

points to an image containing two real images, one cylinder and one sphere, which are obtained from the Michigan State University. The total number of points is 16225. The results by the FRHT method are detected in 198.6 seconds and displayed in Figure E.5 in which some points on the sphere are not identified, whereas the results by the integrated clustering algorithm are detected in 348.4 seconds and displayed in Figure E.6 in which the two clusters look in a better match with the original data image.

Figure C.13 shows an artificially generated image of 200x200 pixels, with three clusters, one sphere, one cylinder, and one frustum of a cone. There are 4062 points in the image. While the depth values are recorded in floating numbers, the regular FRHT method can easily detect all clusters as shown in Figure C.14. While the depth values are integers, the maximum distance error for a point to its right cluster surface becomes one unit of length and the FRHT method does not work well. In this case, since the radius of the cylinder is only 17 units of length, the image can be considered as a coarse or noisy one that is shown in Figure E.7. The integrated algorithm can detect all three clusters in 24.3 seconds and the results are shown in Figure E.8.

Figure E.9 is a more complicated image containing five clusters, one plane surface, one cylinder, one cone, one sphere and one ellipsoid. There are 11038 points in the image and the depth values are recorded in integer numbers. We performed a series of experiments using the integrated clustering algorithm for it.

First, with the approximate distances in FRHT process and the algebraic distances in NFCQS process, the integrated clustering algorithm detected all five clusters as shown in Figure E.10 in 77.8 seconds on a Sparc 10 Station, where the image was divided into

4x4 regions, the minimum number of points for a cluster was set as 9% of the total number of points and the distance threshold was 1.0 unit. The multiple NFCQS processes were used and Figure E.11 shows how the cone cluster is refined by NFCQS. Figure E.11a shows the result of cone cluster by FRHT, E.11b and c are two intermediate results, and E.11d is the final result using NFCQS.

When 50% of total number of points was randomly sampled before the integrated algorithm was applied, the CPU time became 29.2 seconds that is less than the time used by the 100% original data image. However, the CPU time does not seem to be directly proportional to the number of points. Further observing the clustering procedure, we found that the 100% image used 18 NFCQS processes and performed 12638 times of mappings, and the 50% image used 14 NFCQS processes and performed 6134 times of mappings. Moreover, the NFCQS process usually needs more iterations when there are more points being processed. Nevertheless, the more points are taken into process, the more precise the results are. In this example, 1.94% amount of points were missed out for the 50% image and 1.51% amount of points were missed out for the 100% image.

We also used this data set to test the integrated algorithm with exact distances in the FRHT operations. The CPU time was 1326.7 seconds that was much more than the time in above experiments, but the result was more precise and only 0.00453% amount of points were missed out.

For the range image clustering problem, Krishnapuram et al. [1993b] propose the Unsupervised Quadric Surface Fitting (UQSF) algorithm. In the algorithm, the range image is first sampled so that the number of points to be processed is reduced. Then a

planar approximation of the data is obtained by the Compatible Cluster Merging algorithm. In the next step, the neighboring planes are merged to produce quadric surfaces by the Quadric Compatible Cluster Merging algorithm. Then the Possibilistic C Plano-Quadric Shells algorithm is used with all data points to refine the results. Because the intensive computation has to be conducted for all points iteratively in each step of the algorithm, the algorithm might be time consuming. As an example, for a 200x200 image with three or four clusters, the authors stated that the CPU time spent by the algorithm on a SPARC 1 Station was between 30 to 60 minutes. In order to compare the computation speed between the Unsupervised Quadric Surface Fitting algorithm and the integrated clustering algorithm, the range image in Figure E.12 with three clusters is used for the experiment. It is a computer generated range image for a lamp, its depth values are recorded with integers and the total number of points is 13065. The CPU time spent by the integrated algorithm on a SPARC 10 Station is only 45.4 seconds that is roughly equal to 450 seconds on a SPARC 1 Station if we assume that Sparc 10 is ten times as fast as Sparc 1. The results are shown as in Figure E.13. It is noted here that the integrated algorithm uses all the points while the Unsupervised Quadric Surface Fitting algorithm uses only 20% of the points. If 50% of total number of points is randomly sampled before the integrated algorithm is applied, the CPU time becomes 38.5 seconds. Therefore, the integrated clustering algorithm possesses faster computation speed than the UQSF algorithm. Since the UQSF algorithm is based on first fitting planes, it is easy to do so for the shapes like cylinders and cones. But it may not work well for examples like Figure C.1, because each cluster may require too many planes.

A common disadvantage of most clustering algorithms is that the number of clusters needs to be decided in advance. The integrated clustering algorithm proposed in this chapter combines the advantages of the FRHT method and the NFCQS algorithm to form a robust clustering algorithm with which the number of clusters is not necessary to be determined in advance, the detection accuracy will be improved and the computation speed might even be increased. It is particularly suitable for the images containing noise points or outliers. This algorithm can be easily extended to other situations.

Table 5.1 The Numerical Results of the Image Shown in Figure C.5 by the FRHT Method with 3x3 Regions.

Cluster	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
1	0.0556	0.1250	0.0	0.0	0.0	0.0	2.2225	5.0003	1.0000	68.227
2	0.0556	0.1250	0.0	0.0	0.0	0.0	2.2223	0.0	1.0000	18.223
3	0.0556	0.1250	0.0	0.0	0.0	0.0	0.0	0.0	1.0000	-4.000
4	0.0556	0.1244	0.0	0.0	0.0	0.0	-2.218	-0.008	1.0000	18.147
5	0.0556	0.1252	0.0	0.0	0.0	0.0	-2.224	-0.003	1.0000	18.247

Table 5.2 The Numerical Results of the Image Shown in Figure C.5 by the Integrated Clustering Method.

Cluster	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
1	0.0556	0.1250	0.0	0.0	0.0	0.0	2.2222	4.9999	1.0000	68.221
2	0.0556	0.1250	0.0	0.0	0.0	0.0	2.2226	0.0	1.0000	18.226
3	0.0556	0.1250	0.0	0.0	0.0	0.0	0.0	0.0	1.0000	-4.000
4	0.0556	0.1252	0.0	0.0	0.0	0.0	-2.222	0.0	1.0000	18.222

Table 5.3 The Numerical Results of the Image Shown in Figure C.11 by the Integrated Clustering Method.

Cluster	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
1	0.0554	0.1248	0.0	0.0	0.0	0.0	2.2151	4.9927	1.0000	68.088
2	0.0552	0.1249	0.0	0.0	0.0	0.0	2.2101	0.0	1.0000	18.113
3	0.0553	0.1248	0.0002	0.0002	0.0002	0.0	0.0020	0.0011	1.0000	-3.993
4	0.0556	0.1250	0.0	0.0	0.0	0.0	-2.224	0.0018	1.0000	18.251

CHAPTER 6

CLUSTER VALIDITY

6.1 Introduction

Clustering techniques are very important tools for data analysis and pattern recognition. To confirm or ascertain the clustering results, many clustering validity criteria have been developed [Dubes and Jain 1979, Dave et al. 1990, Dave 1991b, Brailovsky 1991, Krishnapuram et al. 1993b, etc.]. Normally, the data is checked for clustering tendency before further clustering. Only if the data tends to be non-random a cluster structure is imposed by clustering algorithms. Hence, the clustering validity includes two aspects, one is whether the data is valid or the points are valid, the other is whether the structure or model is suitable for the data [Dubes and Jain 1979]. The following refers them as data validity and cluster validity respectively.

The pattern matrix and the proximity matrix can be used to record and analyze the features of data that might lead to the data validity measurement. In a pattern matrix each pattern is described by a set of measurements or features. Whereas in a proximity matrix each row or column represents a pattern. The entries are index values of similarity or dissimilarity. However, sometimes it is very difficult or even impossible to check the data validity, such as the image data without prior information. The cluster validity is a step right after the model is obtained and is an important tool to ascertain or deny the imposed structure or model of the data. The following will mainly discuss the cluster validity.

6.2 Cluster Validity

The cluster validity measuring methods are categorized as global validity measures and individual validity measures [Dave *et al.*1990a, Dave 1991b, Krishnapuram *et al.* 1993b]. The global validity measures are used to judge the overall partition of the data set into all clusters. The individual validity measures are used to evaluate the goodness of a particular cluster.

6.2.1 Global Validity and Individual Validity

With the fuzzy clustering algorithms, all data points are assigned fuzzy partitions against all clusters. The amount of fuzziness usually reflects uncertainty. The data points with the greater memberships possess the less uncertainty associated with their classification. From the membership updating methods, we can see that the distances between different clusters do affect the uncertainty. Therefore, the data points within well separated clusters have greater memberships to their clusters and possess less uncertainty. The goal of global validity measures is to find a criterion to measure the classification uncertainty. Various measures have been proposed, such as the partition coefficient [Bezdek 1974], the fuzzy set decomposition measure [Backer and Jain 1981], the classification entropy [Bezdek 1981], the proportion exponent [Windham 1981], and the polarization degree [Dimitrescu 1988]. Since these measures are based solely on the fuzzy partition matrix, they lack direct connection to the geometric properties of the clusters and they do not work well in the case of shell type clusters [Dave 1991b, Krishnapuram et al. 1993b]. Some measures include the nature of the geometry of the data set, such as the sum-of-squared-errors

criterion [Thorndike 1953], the related-minimum-variance criterion [Duda and Hart 1973], the figure of merit [Hoffman and Jain 1987] and the hypervolume and density criterion [Gath and Geva 1989]. However, these measures are used to deal with the compact or filled clusters.

For the shell type clusters, Dave [1991b] introduced a measure based on the hypervolume and partition density for the case of spherical and elliptical shells. It is generalized to the general shell types as the following [Krishnapuram et al, 1993b].

Using the same terminology defined in Chapter 3, the fuzzy shell covariance matrix can be defined as

$$C_{S_i} = \frac{1}{N_i} \sum_{j=1}^{n} (u_{ij})^m \tau_{ij} (\tau_{ij})^T, \tag{6.1}$$

where τ_{ij} is the distance vector from a point v_j to a shell prototype p_i, and

$$N_i = \sum_{j=1}^{n} (u_{ij})^m . \tag{6.2}$$

The shell hypervolume of a cluster can be defined as

$$V_{S_i} = \sqrt{\det(C_{S_i})} \quad . \tag{6.3}$$

The shell density of a cluster may be defined as

$$D_{S_i} = \frac{S_i}{V_{S_i}}, \tag{6.4}$$

where S_i is the sum of close members of shell p_i given by $S_i = \sum_j u_{ij}$ such that $\tau_{ij}^T C_{S_i}^{-1} \tau_{ij} < 1$. The total shell hypervolume V_S and shell partition density D_S may be defined as

$$V_S = \sum_{i=1}^{c} V_{S_i} \quad , \tag{6.5}$$

$$D_S = \sum_{i=1}^{c} D_{S_i} \, . \tag{6.6}$$

Dave et al. [1990a] also introduces T, the thickness of a spherical shell to measure the global validity,

$$T = \frac{\sum_{i=1}^{c} [\sum_{j=1}^{n} (u_{ij})^m (\|v_j - c_i\| - r_i)^2 / N_i]}{[\sum_{i=1}^{c} r_i] / c}, \tag{6.7}$$

where c_i and r_i represent the center and the radius of a cluster respectively. Krishnapuram et al. [1992 and 1993c] propose similar measures. The fuzzy average shell thickness of a cluster is defined as:

$$T_{S_i} = \frac{1}{N_i} \sum_{j=1}^{n} (u_{ij})^m \|\tau_{ij}\|^2 . \tag{6.8}$$

Then the total fuzzy average shell thickness T_S is defined as:

$$T_S = \sum_{i=1}^{c} T_{S_i} \, . \tag{6.9}$$

The global validity measures are used to evaluate the overall partition of the data set, which is very time consuming and is not guaranteed to work well for noisy or complex data sets.

The individual validity measures are used to ascertain or deny the results for a single cluster, and are therefore computationally more efficient. The previously defined measures, such as the shell hypervolume of a cluster in (6.2), the shell density of a cluster in (6.3) and the fuzzy average shell thickness in (6.7), may be used to measure the

individual validity for shell clusters. Their reliability will be tested and compared in the later experiment.

6.2.2 Surface Density

In fact, the shell clusters lie in the subspaces of feature space. A more appropriate validity measure for such subspace clusters should take into account the density of the cluster. The following surface density criterion [Krishnapuram et al, 1993b] is a valuable validity measure for shell clusters that is related to the density of the cluster.

The surface density criterion measures the number of points per unit curve length or surface area instead of the number of points per unit shell volume, which takes into account the fact that shell clusters are subspace clusters. In the two-dimensional case, the surface density δ_i of cluster p_i is defined as the number of points per unit estimated arc length:

$$\delta_i = \frac{S_i}{2\pi r_{effi}} \quad . \tag{6.10a}$$

In the three-dimensional case,

$$\delta_i = \frac{S_i}{4\pi r_{effi}^2} \quad . \tag{6.10b}$$

In (6.10a) and (6.10b), S_i is the sum of central members defined as

$$S_i = \sum_{j \in J} u_{ij}, \quad J = \{k | 1 \le k \le n, \|\tau_{ik}\| < \tau_{\max}\}, \tag{6.11}$$

and $\tau_{\max}$ is the threshold distance, only the points that do not lie too far from the shell are counted. In Equations (6.10a) and (6.10b), r_{effi} is the effective radius of shell cluster p_i

defined as

$$r_{effi} = \sqrt{Tr(C_i)}, \tag{6.12}$$

where C_i is the covariance matrix of cluster p_i, which is different than the fuzzy shell covariance matrix C_{S_i} in (6.1) for shell clusters.

Assume that the circular arc in two-dimensional space is perfect and continuous, the theoretical value of surface density for the arc can be derived as [Krishnapuram *et al.* 1993b]

$$\delta_{arc} = \frac{\gamma}{2\pi\sqrt{1-\frac{4\sin^2(\gamma/2)}{\gamma^2}}}, \tag{6.13}$$

where γ is the angle of the arc. Equation (6.13) indicates that δ_{arc} is independent of radius of the arc. Using (6.13), the theoretical value of surface density for a complete circle, a semi-circle and a quarter circle can be obtained respectively as 1.0, 0.65 and 0.57. Hence, in order to make surface density invariant to partiality, a compensation factor is introduced as [Krishnapuram *et al.* 1993b]

$$f_p = \begin{cases} 1.74 - 0.172 \times (\frac{r_{effi}}{r} - 0.44) & \frac{r_{effi}}{r} \in [0,0.44) \\ 1.54 - 0.594 \times (\frac{r_{effi}}{r} - 0.77) & \frac{r_{effi}}{r} \in [0.44,0.77) \\ 1.0 - 2.34 \times (\frac{r_{effi}}{r} - 1.0) & \frac{r_{effi}}{r} \in [0.77,1.0] \end{cases} \tag{6.14}$$

where $\frac{r_{effi}}{r}$ can be computed before the computation of cluster validity. Actually, $\frac{r_{effi}}{r}$ can be taken as a measure of partiality for circular clusters. Its value indicates the partiality, and for a complete circle, the value is 1.0.

After the scaling factor is introduced, the surface density for a circular arc becomes

$$\delta'_{arc} = f_p \delta \quad . \tag{6.15}$$

The above discussed surface density criterion is for measuring the individual validity, but it can also be used as a global validity measure to evaluate the overall partitions when the surface densities for all clusters are summed up.

However, this measure is more suitable when it is expected that circle boundaries in an image is one pixel thick and more or less continuous. It can not distinguish between the following two cases shown in Figure 6.1. The next section will propose a new measure handling this problem.

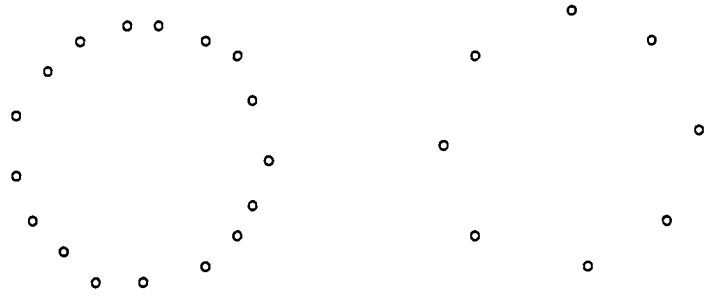


Figure 6.1 Two Circles with Different Distributions of Points.

6.3 A New Validity Criterion for Circular Clusters

In practical applications, the points on a circular cluster are not always distributed equally along its circumference due to various reasons, and the circular images might not be

perfect and continuous. As a common sense, a partial circle should possess a smaller validity value than a complete one, and a circle with uneven-distributed points should have a smaller validity value than a circle with well-distributed points. The following will propose a new validity criterion to numerically evaluate the circular images.

6.3.1 Eccentricity

Before further discussion, it is necessary to introduce the concept of average mass center that is defined as:

$$C_m = \frac{\sum_{i=1}^{n} w_i v_i}{\sum_{i=1}^{n} w_i}, \tag{6.16}$$

where C_m is the vector of average mass center of n points, v_i and w_i are respectively the vector and the weight factor of ith point.

As we know, a complete circle is a symmetrical shape, and its geometrical center and its average mass center should coincide. On the contrary, the geometrical center and the average mass center for an incomplete circle might not coincide. Hence, the difference between both centers might be used as a parameter to measure how complete a circle is. Based on this idea, a concept of eccentricity is given as below.

Eccentricity: The eccentricity of a circle, E_c is defined as:

$$E_c = \frac{D_c}{R}, \tag{6.17}$$

where D_c represents the distance between the geometrical center and the average mass center of the circle, and R is the radius.

Generally, the eccentricity is a value between 0 and 1 for a circular shape if all points are located on the circular arc. Using this definition, a few examples, such as a complete circle, a half circle and a quarter circle, are listed in Table 6.1. The eccentricity values show us how symmetrical the circles are to their geometrical centers.

6.3.2 Distribution Measurement

Usually, the points on a circular image are unevenly distributed along its arc, namely, there are more points on some arc sections but fewer points or even none on other sections. If there are same number of points on two circles, the circle with well-distributed points should be considered as a better cluster than the one with uneven-distributed points. A measurement of distribution is proposed as the following.

Divide the circle equally into N_s sections, and find the average number of points in each section, n_{ave}, by

$$n_{ave} = \frac{n}{N_s}. \tag{6.18}$$

Then check every section in turn to see how many points it contains. If the number of points is more than the average number of points on a section, the section is counted as a valid section. After all sections are checked over, the number of valid sections, N_{vs} is obtained. As the last step, the distribution measure Π is computed by

$$\Pi = \frac{N_{vs}}{N_s}. \tag{6.19}$$

To sum up, the method of distribution measure computation is briefly summarized as below.

(1) Divide the circle into N_s sections and find the average number of points on each section n_{ave} by (6.18);

(2) Count the number of valid sections N_{vs};

(3) Compute the distribution Π by (6.19).

It is easy to see that the distribution is also a value between 0 and 1. The results for a few examples are listed in Table 6.1, which shows their distribution measures.

6.3.3 Valid Points and Valid Density

The above defined concepts of eccentricity and distribution can denote different aspects of the circle validity. A synthesized parameter, valid points V_p is defined as

$$V_p = n\Pi(1-E_c) \ , \tag{6.20}$$

which reasonably reflects the effects of eccentricity and distribution. However, it can not exactly represent the validity as it is affected mostly by the total number of points, in other words, it should be related with the number of points per unit arc. The following proposed valid density, V_d is defined to meet the requirement.

Definition: The valid density of a circle is the valid points per unit arc that can be represented as

$$V_d = \frac{V_p}{2\pi R} \quad . \tag{6.21}$$

From the above definition, we can see that a smaller circle has a bigger valid density value than a bigger circle when they contain the same number of points, which makes the criterion fair. Nevertheless, it should be pointed out that the valid points V_p and the valid density V_d might not be integers. Table 6.1 shows a few examples.

6.3.4 Conclusions

The above discussion is based on the case that all points are located right on the circular arc. Unfortunately, some of points might not lie on it in practical images. In that case, it is suggested to merely compute the valid density using those points that do not lie too far from the arc. Such requirement is comparable to use of "central members" by other validity measures.

Sometimes, a circular image becomes incomplete due to camera position or frame position, such as the edge images of the spheres located close to image frame edge in Figure 1.1. It might not be fair to these circles if the same method as in Equation (6.20) is used to compute their valid densities. It is necessary to compensate the valid density for these circles as below.

First it is needed to count N_{bs}, the number of blank sections that do not contain

any points. Then the amended valid density V_d' is obtained by

$$V_d' = \frac{V_d}{(1-\frac{N_{bs}}{N_s})(1-E_c)\Pi} \quad . \tag{6.22}$$

A few examples are listed in Table 6.1, which indicate the effect of the amended method. From the table, we can see that with the same amount of points, the complete circle with radius R=10 has the same amended valid density as the half circle with radius R=20.

For those examples in Table 6.1, where all points are exactly located on the boundary and have hard memberships, the measures of shell hypervolume, shell density and shell thickness become meaningless. The validity values of surface density for those examples are calculated and listed in Table 6.1. From the table, we can see that the surface density is independent of radius and number of points, i.e., the same type of circles has the same value of surface density no matter what their radii and number of points are. The amended surface density is one for all shapes. While applying the valid density measure, we can identify them by different values, for instance, with the same radius and number of points, a complete circle has a greater validity value than a half circle and a half circle has a greater validity value than a quarter circle; with the same radius and shape, a circle with more points has a greater validity value than the one with fewer points. After the amended valid density measure is applied, the clusters with the same radius and actual distribution of points have a same validity value, for example, a half circle with 360 points has the same validity value as the complete circle with 720 points when both of them have the same radius. Therefore, the valid density measure is more reasonable measure considering the distribution situation and the eccentricity of a circular cluster.

In Chapter 7, a practical example is used to demonstrate the applicability of different validity criteria presented here. Though the criteria proposed here only focus on the circular shapes, they can be extended to other two-dimensional shapes and three-dimensional shapes, which is the future work.

Table 6.1 The Examples for Valid Density Criterion ($N_s = 36$, and all points are well-distributed on the specific arc section).

Circle	Complete			Half			Quarter		
n	360	360	720	360	360	720	360	360	720
R	10	20	20	10	20	20	10	20	20
Ec	0.000	0.000	0.000	0.64	0.64	0.64	0.9	0.9	0.9
Π	1.000	1.000	1.000	0.500	0.500	0.500	0.250	0.250	0.250
Vp	360	360	720	64.8	64.8	129.6	9	9	18
Vd	5.73	2.86	5.73	1.04	0.52	1.04	0.14	0.07	0.14
Vd'	5.73	2.86	5.73	11.46	5.73	11.46	22.92	11.46	22.92
δ	1.0	1.0	1.0	0.65	0.65	0.65	0.57	0.57	0.57
δ'	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

CHAPTER 7

PRACTICAL APPLICATION TO THE GRANULAR FLOW RESEARCH

7.1 Introduction

Motion analysis using vision based methods has found many practical applications in a variety of engineering problems, ranging from automatic target detection for defense, to human motion analysis in human factor and bio-mechanical engineering. Recently, there has been a considerable increase in the use of such techniques for experimental studies in multi-phase flows [Ghidaglia et al. 1993, Hsiau and Hunt 1993, Krishnan et al. 1992, Li et al. 1992]. These applications include measurement of velocity fields, measuring flow patterns, describing the flow structure (including micro-structure of particles), and finding velocity distribution of the particles. In general, the image analysis is based on simple techniques such as binary image analysis, histogram analysis of gray level images, or template matching techniques [Ghidaglia et al. 1993, Hsiau and Hunt 1993, Krishnan et al. 1992, Li et al. 1992]. As the applications are becoming more complex, accurate shape detection in relatively low resolution, noisy images is required for obtaining meaningful results [Hanes 1993, Dave et al. 1993]. In such situations, simpler approaches such as binary image analysis do not work well. As a consequence, the task of image processing, which was once considered a simple exercise, requires considerable amount of research and/or development.

7.2 Physical Experiment

The physical experiment used as an example here comes from the experimental studies of granular flows. Research in that field is relevant to many areas, such as industrial solids handling, plastics processing, pharmaceuticals, and natural granular flows such as avalanches and land-slides. Lack of reliable theories has led to numerous experimental investigations [Dave et al. 1993, Foerster et al. 1994, Ghidaglia et al. 1993, Hanes 1993, Hsiau and Hunt 1993, Li et al. 1992, Yu 1993], from which an example is drawn here. The experiment involves measurement of particle configurations in inclined chute flows [Hanes 1993], or random packing of the spheres [Rosato 1993]. It is described below with the details of image analysis requirements.

Locating and tracking individual particles is often the main objective of many experimental studies in granular flows. If the image quality is good and the scene is not too complex, simple binary analysis or gray level template matching techniques can be used successfully. Unfortunately, many times the image quality may not be good, and the patterns may be only partially visible. An example of closely packed spheres is shown in Figure 1.1, where the objective is to locate as many spheres as possible, with reasonable accuracy. This task requires that the method used is, a) fast since there are many spheres to be detected, and b) reliable for detecting even partially visible spheres.

7.3 Divide and Conquer-Noise FCS Algorithm and Its Application

Consider a practical application, where an image of 256×256 contains circular shapes of diameters ranging from 10 to 60 pixels. For an accurate location of the shapes, detecting

centers to within one pixel may not be enough. However, even for just one pixel accuracy in computing centers as well as radii, a three dimensional array of $256 \times 256 \times 60$ may be required by the conventional HT method. For a sub-pixel accuracy the array would be even larger. As the size of array goes up, so do computations, both in terms of filling the accumulator and then searching for the peaks. An interesting variation of the HT, called the Adaptive Hough Transform (AHT) [Illingworth and Kittler 1987], may be used where the resolution is varied from coarse to fine to achieve higher accuracy without using a very large accumulator array. In theory, one can achieve a very fine resolution and hence high accuracy in object location using this approach. However, in practice, the results are not always as promising.

For the application considered here, the basic HT method was applied without the use of gradients, and the HT method similar to the one in [Kimme et al. 1975] was applied with the use of gradients. In either case, it was observed that the method required a large number of computations to achieve a reasonable level of accuracy. On the other hand, if only a rough estimate of the centers is required then either method is reliable and very quick.

As discussed in Chapter 4, the noise FCS algorithm is robust against noise but needs good initial fuzzy partitions for all data points. The following will propose an integrated algorithm that uses HT method to obtain good initialization for the noise FCS algorithm, which is called Divide and Conquer-Noise FCS (D&C-NFCS) algorithm.

7.3.1 The Algorithm

The proposed algorithm is based on obtaining approximate results using the HT method and then using that information in noise FCS (NFCS) algorithm. Our experience with using HT with a coarse resolution suggests that one can successfully locate most of the nearly circular shapes in the data. When edge gradient information is not available, the circle is parameterized by center coordinates (a,b), and radius r using the following equation

$$(x-a)^2+(y-b)^2=r^2 \tag{7.1}$$

and a three dimensional accumulator array. The resolution of the array is chosen such that each parameter is resolved to an accuracy about six pixels. Thus the accumulator array is coarse corresponding to the accuracy of six pixels as compared to the "fine" array corresponding to 1/4 pixel required accuracy. All the significant peaks above a certain threshold, which is chosen based on available prior knowledge and keeping in mind the possibility of partial shapes, are detected. For each peak, the set of parameters (a,b,r) are used to select all the points within a small neighborhood of the circle of radius r centered at (a,b). This set of points are then processed using NFCS algorithm with radius r and center (a,b) used as initial known parameters for the iterative algorithm to detect a single cluster. It should be noted that even the elliptical shapes are detected using this approach without having to use expensive HT technique for ellipse detection, which is discussed by Dave and Bhaswan [1992].

When edge gradient information is available, we employ a scheme similar to the one in [Kimme et al. 1975]. The circle detection is done in two passes. In the first pass, centers

are detected using a two dimensional array, where the accumulation is done by varying radius r over the expected range, and then computing (a,b) using the following equations.

$$\begin{cases} a = x + (sign) r \cos\varphi \\ b = y + (sign) r \sin\varphi \end{cases} \tag{7.2}$$

Here φ is the gradient angle returned by the edge operator, and the *sign* is set to either + or - depending on the intensity transition across the edge, i.e. whether one crosses from higher to lower or lower to higher intensity as one goes away from the circular object. For the second pass, for each peak of the first pass, only a one dimensional accumulator array is used and by fixing (a,b), values for r are accumulated. In situations where all the objects to be detected have about the same radius, as is the case for the second physical experiment, this method becomes easier, as the second pass is not required. For this method also, the resolution of the accumulator is coarse. Using either of the above HT methods, one can obtain rough estimates of (a,b) and r. Since good estimates for the center and radius are available, the NFCS algorithm stated in Chapter 4 becomes quite simple, in which the number of clusters is set as one, and the initial fuzzy partition matrix is obtained from (a,b) and r.

The whole divide and conquer algorithm is briefly described as below.

Divide and Conquer-NFCS Algorithm:

(1) Set maximum number of clusters, then use conventional HT method to search centers and locations of all possible clusters. According to the different physical constrains of applications, some clusters might be consolidated.

(2) Refine the parameter results of step (1) by NFCS algorithm.

(3) Remove false clusters through checking physical constrains and validity for all detected clusters.

In this algorithm, the NFCS algorithm becomes one of the steps in the overall procedure. A clean-up step is also required so that spurious patterns may be deleted or multiple patterns may be consolidated. This step requires a set of validity and checking criteria, which is mentioned in Chapter 6.

7.3.2 Practical Examples

The above algorithm is applied to the particle packing experiment, where the image of Figure 1.1 is analyzed to locate as many spheres as possible. It is emphasized that the objective is not to find every sphere but to locate as many spheres as possible, in particular all the fully visible spheres are expected to be located. The edge data for this image is shown in Figure F.1. This is the edge map of the original image (about 150×130 pixels) and there are 2596 number of points in the edge data. As can be seen in the figure, the edge data contains some artifacts due to shadows and poor image quality. There are many partial shapes due to hidden geometry. This example appears to be more complex as compared to the examples reported before [Dave and Patel 1990, Krishnapuram, Frigui and Nasraoui 1992, 1993, Krishnapuram, Nasraoui and Frigui 1992]. This data is used in D&C-NFCS algorithm with maximum number of clusters set to 26. Since all circles have a fixed radius, the multiple clusters should be consolidated by the physical constrain that the

distance between any two sphere centers ought to be greater than the radius. The HT accumulator array used in this example is approximately 25×20. The λ used in noise distance was set at 0.5. Validity measures used include shell thickness, fuzzy hyper-shell-volume, fuzzy hyper-shell-density, surface density, number of points per cluster, and valid density. For this experiment, all the circles should be of the same size, so one can restrict the range of radius r in equation (7.2). The results for these conditions are shown in Figure F.2 where the circles are plotted over the edge data, and in Figure F.3 where the circles are superimposed over the image data. These results do not include the effect of applying validity criteria or cluster merging. As can be seen, most circles are found with a very good detection accuracy. However, some of the circles detected are slightly misplaced or even spurious. After all, these results are a significant improvement over the results of HT alone. Several examples of individual clusters are shown in Figure F.4 to demonstrate the improvements made through NFCS algorithm in localization of clusters. In this figure, the results of D&C-NFCS are shown using a solid line circle, and the results of HT alone are shown using a dashed line circle.

7.3.3 Results of Validity Measures and Conclusion

Validity measures for the clusters shown in Figures F.2 and F.3 are presented in Table 7.1. The last six rows present several statistical parameters for each measure. These parameters are: average, standard deviation, maximum, minimum and average + or - one standard deviation of the sample. One may use the last parameter as an indicator of the goodness of fit. For example, in case of shell thickness, clusters having values above 0.091 are questionable, while in case of surface density, clusters having values less than 0.397 are

questionable. Using shell thickness, clusters 14, 15, 17, 20 and 21 are classified as bad clusters. Cluster 12 is a spurious cluster and it is just missed being classified as a bad cluster, while cluster 17 is a good cluster and it is classified as a bad cluster using this criterion. It is interesting that cluster 17 is classified as a bad cluster by all the measures listed. Careful observation of the edge data for this cluster reveals that it is essentially a very short circular arc, and therefore may have been mis-classified.

Table 7.1 Validity Parameters for the Clusters Shown in Figure F.3.

Cluster Number	Shell Thickness	Surface Density	Central Members	Fuzzy Shell-Volume	Fuzzy Shell-Density	Valid Density
0	0.061	0.949	107	10.01	9.47	0.595
1	0.050	0.952	107	8.07	11.24	0.703
2	0.051	0.956	107	8.22	11.47	0.649
3	0.057	0.896	102	9.41	9.34	0.515
4	0.059	1.031	116	9.64	10.66	0.469
5	0.071	0.762	84	10.77	6.73	0.403
6	0.079	0.788	90	12.76	5.90	0.523
7	0.050	0.575	63	7.77	6.22	0.767
8	0.052	0.943	106	8.38	10.96	0.643
9	0.065	0.887	100	10.46	8.39	0.606
10	0.071	0.937	107	11.92	8.12	0.622
11	0.078	0.543	60	11.73	4.32	0.333
12	0.089	0.584	67	13.33	4.44	0.322
13	0.076	0.458	51	11.45	3.95	0.696
14	0.112	0.426	50	18.17	2.56	0.189
15	0.106	0.348	40	13.73	2.71	0.121
16	0.076	0.804	92	12.52	6.37	0.395
17	0.097	0.330	37	14.78	2.30	0.131
18	0.062	0.534	57	9.26	5.14	0.933
19	0.042	0.415	46	6.13	6.00	0.685
20	0.092	0.397	45	11.71	3.52	0.159
21	0.098	0.344	33	13.33	2.28	0.600
22	0.069	0.431	49	11.40	3.88	0.944
23	0.074	0.436	51	13.01	3.60	0.596
24	0.088	0.363	41	10.96	3.28	0.124
25	0.063	0.460	52	10.37	4.40	0.946
Average	0.073	0.637	72	11.13	6.05	0.526
Standard Dev.	0.018	0.240	27	2.49	2.94	0.250
Minimum	0.042	0.330	33	6.13	2.28	0.121
Maximum	0.112	1.031	116	18.17	11.47	0.946
Avg.+/-sd.dev.	0.091	0.397	44	13.62	3.11	0.276

The cluster merging criterion removes the cluster 12 and merges it into cluster 16. The final results of the D&C-NFCS algorithm after cluster removal and merging are shown in Figure F.5. The results shown here indicate that the strategy outlined here works well for such examples. Regarding the validity measures used, although each one results in a few mis-classifications, in general, all the validity measures have worked reasonably well. The surface density measure has one very good feature, i.e. its ability to provide an absolute measure. It gives a value of close to 1 for very good clusters, however, it is not clear what should be the cut-off value for the bad clusters. Using the strategy proposed here, it indicates that clusters 15, 17, 20, 21 and 24 should be thrown out. This results in mis-classification of bad clusters 12 and 14, and good clusters 17 and 24. However, since these clusters are border-line cases, such mis-classification is not entirely unexpected.

While in case of valid density, clusters 7, 13, 18, 19, 21, 22, 23 and 25 are compensated as they locate in close distances to the frame edge. As can be seen from Table 7.1, clusters 14, 15, 17, 20 and 24 should be thrown out using the proposed strategy. Further observation of the table, if a threshold value of 0.350 is applied, then clusters 11 and 12 would also be thrown out. The results are obtained without using merging criterion, and displayed in Figure F.6. Though the occluded good clusters 11, 17 and 24 are mis-classified, all spheres in the top layer are correctly detected.

The final outcome of the D&C-NFCS algorithm is shown in Figure F.5 or F.6, and the computational cost of the whole algorithm to achieve these results was only about 50 seconds of CPU time (on Sparc 10 model 41, one CPU), including the HT part. The reason for its speed is mainly due to the use of segmented data in NFCS algorithm.

Although the unsupervised algorithm similar to the one in [Krishnapuram et al. 1992, 1993c] is not run with this data, one can estimate the time required in such an approach by just running the NFCS on the whole data with about 40 clusters. This is because in order to find about 26 clusters, one might have to use at least 40 clusters to begin with so that after cluster removal and merging, about 26 clusters may be found. It is necessary to point out that the NFCS algorithm is from Dave and Bhaswan [1992], requiring solution to non-linear equations. Therefore it may be slower as compared to quadric shells type algorithms used by Krishnapuram et al. [1992, 1993c]. However, in the comparison presented here, both the D&C-NFCS and the regular NFCS algorithms use the same code to solve the same set of non-linear equations. In order to obtain a fair estimate of time, results of HT were used to initialize the 40 cluster prototypes. Using such initialization, NFCS is run for 15 iterations only and terminated. At that time only some of the clusters were found correctly, and it required over 600 seconds of CPU time. One can realize that this time is still only a fraction of the time required for the complete unsupervised algorithm. Through this simple example, it can be seen that using the proposed technique, one gets an algorithm that is at least an order of magnitude faster than the conventional algorithms.

In Chapter 5, the Integrated Clustering algorithm for general quadric surfaces is proposed, and certainly, a similar algorithm can be developed for 2D images. As to the 2D circular shapes, three independent parameters can determine their model equations, hence only three non-collinear points need to be picked for each mapping. For the image in Figure 1.1, with the restricted radii, the integrated RHT-NFCS method detected the first 30 clusters and then merged them into 11 clusters in 77.2 seconds. The results are shown

in Figure F.7 and F.8. In this case, the advantage of the RHT method seems not so distinguishable as there are a lot of false iterations with it, and on the contrary, the normal HT method based on Equation (7.2) avoids the combination explosive problem by restricting the radii and achieves a faster speed.

CHAPTER 8

THEORETICAL ANALYSIS OF DIFFERENT CLUSTERING METHODS

The Hough based methods and fuzzy techniques have been discussed in early chapters, and their applications to 2D and 3D images have also been presented. The goal of cluster detection can be considered as fitting model equations to image points. Besides the discussed methods, there exist many other fitting methods, such as least squares regression method, least median squares method. This chapter will perform some analysis and comparison among these different methods.

8.1 Least Squares Regression Method

The purpose of regression analysis is to fit equations to observed variables. The general linear model is [Rousseeuw and Leroy, 1987]

$$y_i = x_{i1}\theta_1 + \cdots + x_{ip}\theta_p + e_i \qquad i = \{1,\ldots,n\}, \tag{8.1}$$

where n is the number of variables (sample size). The variables $x_{i1},\ldots,x_{ip}$ are called the explanatory variables or carriers, whereas the variable y_i is called the response variable. In classical theory, the error term e_i is assumed to be normally distributed with mean zero and unknown standard deviation σ. The goal is to find the vector of parameters

$$\Theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_p \end{bmatrix} \tag{8.2}$$

from the given data:

$$\begin{bmatrix} x_{11} & \cdots & x_{1p} & y_1 \\ \vdots & & \vdots & \vdots \\ x_{i1} & \cdots & x_{ip} & y_i \\ \vdots & & \vdots & \vdots \\ x_{n1} & \cdots & x_{np} & y_n \end{bmatrix}. \tag{8.3}$$

Applying a regression method to the data set yields

$$\hat{\Theta} = \begin{bmatrix} \hat{\theta}_1 \\ \vdots \\ \hat{\theta}_p \end{bmatrix}, \tag{8.4}$$

where the estimated vector, $\hat{\Theta}$ represents the regression coefficients. Then $\hat{y}_i$, the predicted or estimated value of y_i can be obtained by

$$\hat{y}_i = x_{i1}\hat{\theta}_1 + \cdots + x_{ip}\hat{\theta}_p, \tag{8.5}$$

The residual r_i of the ith case is defined as:

$$r_i = y_i - \hat{y}_i. \tag{8.6}$$

To solve the above problem, different regression methods have been developed. The least squares (LS) method is a well-known one that has become the cornerstone of classical statistics. The basic idea is to minimize the sum of squared residuals, which can be represented as

$$\underset{\hat{\Theta}}{Minimize} \sum_{i=1}^{n} r_i^2 \tag{8.7}$$

The LS method can be computed explicitly from the data, but its application is restricted by some kinds of outliers in the data, i.e., it does not work well under some circumstances. For example, Figure 8.1a contains five points, $(x_1, y_1),\ldots,(x_5, y_5)$ in a two-dimensional data set, with a well-fitting LS line. If the point (x_1, y_1) is shifted along

x-axis as shown in Figure 8.1b, then the LS regression line is completely tilted. The resulting point is an outlier which is called a leverage point since its effect on the LS estimator is very large [Rousseeuw and Leroy, 1987].

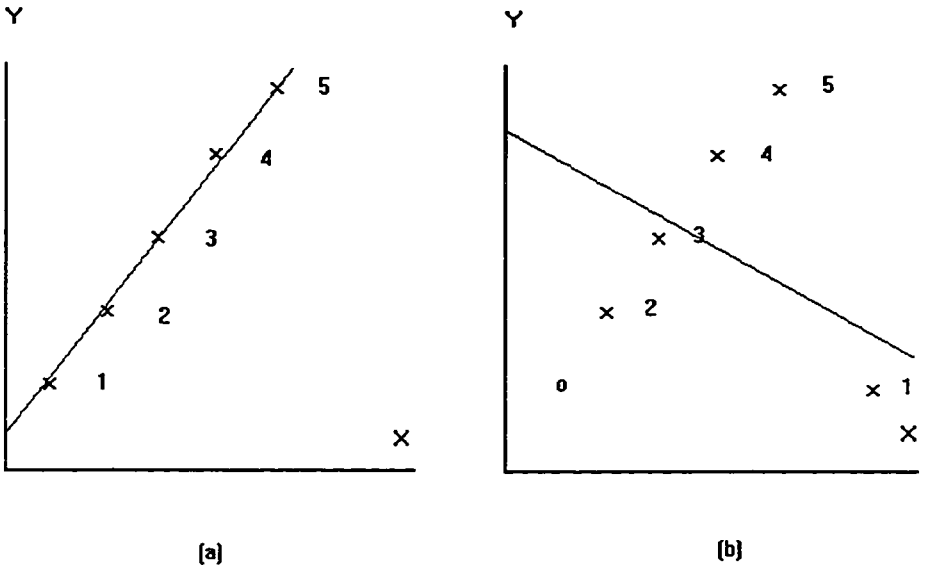


Figure 8.1 LS regression line with respect to the data (a) without outliers, and (b) with an outlier.

Obviously, LS method might not deal with the noise data containing outliers. So the development of robust regression methods has been performed, which tries to devise regression estimators that are not so strongly affected by outliers. Before further discussion of different estimators, it is necessary to introduce a concept, breakdown point, which can evaluate the ability of an estimator [Rousseeuw and Leroy, 1987]. Take any sample of n data points,

$$Z = \{(x_{11},\ldots,x_{1p},y_1),\ldots,(x_{n1},\ldots,x_{np},y_n)\}, \tag{8.8}$$

and let T be a regression estimator, i.e.,

$$T(Z) = \hat{\Theta}. \tag{8.9}$$

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

$$bias(m;T,Z) = \sup_{Z'} \|T(Z') - T(Z)\|, \tag{8.10}$$

where the supremum is over all possible Z'. If $bias(m;T,Z)$ is infinite, this means that m outliers can have an 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 the sample Z is defined as

$$\varepsilon_n^*(T,Z) = \min\{\frac{m}{n}; bias(m;T,Z) \text{ is infinite}\}. \tag{8.11}$$

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

For least squares method, as can be seen from Figure 8.1, one outlier is sufficient to carry T over all bounds. Therefore, its breakdown point equals $1/n$ which tends to zero for increasing sample size n, so it can be said that LS has a breakdown point of 0%. This indicates the extreme sensitivity of the LS method to outliers.

However, based on the LS method, many methods have been proposed to fit curves [Haralick and Shapiro 1992]. It is necessary to point out that those LS based methods might have slightly higher than 0% breakdown points.

8.2 Least Median of Squares

A complete name for the LS method would be least sum of squares, Rousseeuw [1984] proposed a robust regression method, least median of squares (LMS), by replacing the word "sum" with "median". LMS can be represented as

$$\underset{\hat{\Theta}}{Minimize}\ \underset{i}{med}\, r_i^2, \tag{8.12}$$

The breakdown point of the LMS technique is as high as 50%, the best that can be expected by any methods [Rousseeuw and Leroy 1987].

As we know, the LS regression method has a straightforward formula to compute the regression coefficients, which makes it simple to use. However, the LMS technique can not be explicitly represented by a formula. It is implemented as a complete code called PROGRESS [Leroy and Rousseeuw 1984]. The algorithm in PROGRESS proceeds by repeatedly drawing subsamples of p observations, which is similar to the drawing method in the FRHT method. For each subsample, indexed by J, the corresponding vector of coefficients Θ_J can be achieved. Then a median value for the subsample can be obtained by

$$\underset{i=1,\ldots,n}{med}(y_i - x_i\Theta_J)^2. \tag{8.13}$$

Finally, the minimal median value is retained until the drawing is finished.

Theoretically, one could repeat the sampling procedure for all possible samples of size p, of which there are C_n^p. Unfortunately, C_n^p. increases very fast with n and p, which would bring the combination explosive problem to some applications. Therefore, a certain number of random selections is recommended, such that the probability that at least one of the m subsamples contains all good points is close to 1. A subsample containing all good

points means it consists of p good observations of the sample, which may contain up to a fraction ε of bad observations. The expression for this probability, assuming that n/p is large, is

$$1-(1-(1-\varepsilon)^p)^m \quad . \tag{8.14}$$

Obviously, for a specific probability that must be close to 1, the number of subsamples m can be determined for given values of p and ε. Rousseeuw and Leroy [1987] created a table shown in Appendix B, for $p \leq 10$ and for a percentage of contamination varying between 5% and 50%.

Based on the basic idea of LMS, Rousseeuw and Leroy [1987] propose the minimum volume ellipsoid (MVE) estimator, and Jolion et al. [1991] further extend it as a generalized minimum volume ellipsoid (GMVE) clustering algorithm. In section 8.4, the MVE type algorithms will be discussed in detail.

8.3 Comparison among Different Methods

In order to apply the LS and the LMS methods to the data with multiple clusters, one needs to perform a presegmentation to the data set, which is very difficult to accomplish when the data are scattered or noisy, and the overlapping clusters exist. As to the early described Noise Fuzzy Clustering and FRHT methods, they directly identify clusters from the original data instead of performing presegmentation and then merging or fitting.

As discussed in above sections, the LS method has a 0% breakdown point, and the LMS method approaches a 50% breakdown point. The FRHT method should have a 50% breakdown point because of that there is always a solution for the right cluster if the

number of noise points is not over 50%. Full scale analysis in multivariate data in higher dimension is beyond the scope of this project. We can look into a simple example as shown in Figure 8.2, which contains only one cluster and some noisy points. If the number of *good* points in the data set is 50% of the total amount of points, then if the minimum number of points for a cluster is set as a number approaching 50% of the total number of points, the FRHT can always detect the true cluster no matter what kind of noisy points is there. The FRHT uses a similar sampling method as LMS. Both of them should have the same breakdown point, 50%, because theoretically they can take all combinations of subsamples from the data set.

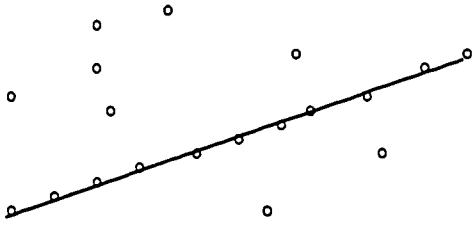


Figure 8.2 A Data Set Containing a Cluster and Some Noisy Points.

As to the Noise Fuzzy Clustering method, with good initial membership partitions and appropriate initial parameters, the breakdown point might approach a higher number. Full scale analysis is beyond the scope of this project. With the same example shown in Figure 8.2, if there is a very good initial membership matrix, the Noise Fuzzy Clustering method can detect the true cluster correctly when there are more than 50% of good points

in the data set. As mentioned before, the Noise Fuzzy Clustering method is sensitive to the initial membership matrix, but theoretically there always exists a good initial membership matrix. Therefore the Noise Fuzzy Clustering method should have a higher breakdown point. The concept of breakdown point is introduced for the case of single cluster where the highest breakdown point is near 50%. For the case of multiple clusters, theoretically the highest breakdown point is still 50%, but in most situations, the clustering algorithms for multiple clusters may not "break down" when the number of outliers is greater than 50%. For instance, when the FRHT is used to detect multiple clusters, for a specific cluster, both of the noise points and the data points belonging to other clusters could be taken as outliers, obviously they might be more than the data points in the detected cluster. Therefore, both of the FRHT and the Noise Fuzzy Clustering methods possess higher breakdown point, which is clearly demonstrated from the examples in the early chapters.

The goal of all the four mentioned methods is to find an equation in a better match for the data points. The LS method and the Noise Fuzzy Clustering method use all data to minimize their objective functions, and there is a closed form solution for the LS method when the function is linear, whereas the Noise Fuzzy Clustering method needs to iteratively update the membership partitions till they become stable, generally, it is computationally expensive. Though the LMS method might not have to use all the data as discussed in Section 8.2, it is necessary to use all the data in order to get a 100% reliable solution, which might bring up the combination explosive problem when the number of points and the number of parameters are greater ones. The FRHT method uses some of

the data instead of all the data to search the right clusters, which makes it a faster detection method.

To sum up, the comparison among different methods is shown in Table 8.1, where NFC stands for Noise Fuzzy Clustering method.

Table 8.1 Comparison Among Different Methods

Method	Pre-segmentation	Breakdown point	Closed Form Solution	Data involved	Robust
LS	Yes	Low	Yes*	All	No
LMS	Yes	High	No	All or partial	Yes
NFC	No	High**	No	All	Yes
FRHT	No	High**	No	Partial	Yes

* Only when the equation is linear. **Refer to the text.

Through the above comparison, we can see that the LMS, the Noise Fuzzy Clustering and the FRHT methods have high breakdown points and are robust, whereas the LS method has a low breakdown point; the Noise Fuzzy Clustering and the FRHT methods can be used to detect multiple clusters from an image data set without presegmentation, but the LS and the LMS can not; the FRHT method randomly picks up some number of points from the data set to fit the data to a equation, whereas the other three methods use every point in the data. It would be an interesting issue to combine their advantages to form an efficient algorithm, for instance, combining the features of high speed in FRHT, multiple clusters and fuzzy partitions in Noise Fuzzy Clustering, and robustness in LMS to form a fast robust clustering algorithm, which is the future work.

8.4 A New Clustering Algorithm for Quadric Surfaces

The minimum volume ellipsoid estimator extracts the cluster with least volume region containing h percent of the data points. The h is usually 0.5, and the MVE can always find the true cluster when the number of outliers is less than 50% for a single cluster data set. For the data set with multiple clusters, the MVE would not work well because in most cases, the number of outliers is more than 50% for a specific cluster. The MVE is extended as the generalized minimum volume ellipsoid estimator that can be used to solve many pattern recognition problems in computer vision [Jolion et al. 1991]. The GMVE searches a cluster at an iteration. Similarly as the FRHT method, when a best cluster is found, the data points belonging to the cluster are removed from the data set and a new iteration begins. The detection process terminates until the number of remaining points becomes less than the minimum number of points a cluster must contain or the number of detected clusters reaches the maximum number of clusters. At each iteration, the MVE estimator is applied with different values of h decreasing from 0.5. For each h, a cluster with minimum volume ellipsoid can be obtained. After all clusters with different values of h are obtained, the Kolmogorov-Smirnov test is employed to check the significance level of all clusters at different h, and the cluster with the smallest significance level over all h values is the best cluster at the iteration.

Just like the FRHT, the GMVE is a kind of "find a cluster at a time" algorithm, and is able to identify all true clusters in the data set. However, because the GMVE needs extensive computations its speed is very slow. Moreover, it can always obtain a solution even when there does not exist a true cluster in the data set.

Rousseeuw [1983, 1984] introduced the least trimmed squares (LTS) estimator for regression, which is given by

$$\underset{\hat{\Theta}}{Minimize} \sum_{i=1}^{h_0} (r^2)_{i:n}, \tag{8.15}$$

where $(r^2)_{1:n} \leq \cdots \leq (r^2)_{n:n}$ are the ordered squared residuals. The best robustness properties are achieved when h_0 is approximately $n/2$, in which case the breakdown point attains 50%.

Based on the above discussion, we can develop a new algorithm called Generalized Least Trimmed Squares for Quadric Surfaces (GLTS-QS) as following for the range image data discussed in the early chapters.

GLTS-QS algorithm:

(1) Set the minimum number of points a cluster must contain, the maximum number of clusters, the maximum number of samplings for an iteration, the distance threshold *Dt* for considering a point belonging to a cluster, the maximum and minimum values of *h*. Initialize the least sum of squared distances at all different *h* as positive infinity (or a positive large number).

(2) For a sampling, randomly sample nine different points and substitute their values into (2.10) to obtain nine equations. Then solve the nine joint equations to obtain the parameters of a possible cluster. Calculate the distances of all remaining points from the possible cluster and sort them by magnitude in a ascending order. Calculate the sum of squared distances of the *h* percent of points with smaller distances for all specific *h* values, and if the sum of squared

distances is smaller than the least sum of squared distances for a specific h, just update it with the new value and store the corresponding parameters; otherwise, continue.

(3) If the number of samplings reaches the maximum number of samplings, continue; otherwise, go to (2).

(4) For a possible cluster with the least sum of squared distances at a h value, check how many points in the current data set are located on the boundary of the cluster, where a point is considered lying on the cluster if the distance of it from the calculated boundary is shorter than Dt.

(5) If the clusters at all different h values are checked, continue; otherwise, go to (4).

(6) Take the cluster with most points on its boundary as the best cluster at the iteration, and remove the points on the cluster from the data set.

(7) If the remaining number of points becomes less than the minimum number of points a cluster must contain, or the number of detected clusters reaches the maximum number of cluster, the clustering process is over; otherwise, go to (2).

In the algorithm, the minimum and maximum values of h can be determined from the information of the data set. The minimum value takes the minimum size of cluster, and the maximum value takes the smaller value from 0.5 and the maximum size of cluster. For example, when the minimum size is 0.1 or 10% of the whole data set and the maximum

size is 0.4 or 40% of the whole data set, the minimum and maximum values of h can take 0.1 and 0.4 respectively. The value of h is decreased from maximum to minimum by a factor of 2. In the above example, the first value is 0.4, the second one is 0.2 and the last one takes 0.1. As to the selection of the maximum number of samplings, one can refer to the analyses in 8.2 for LM, and in 2.2 for RHT. When there are more clusters in the data set of quadric surfaces, an extremely large number is needed to ensure the true cluster can be detected. Obviously, the window method in FRHT can reduce the number to a very small one.

For the image shown in Figure C.13, the GLTS-QS algorithm detected all three clusters correctly in 608.6 seconds on a Sparc 10 Station, where the values of h were set as 0.4 and 0.2, the Dt was set as 1.0, the maximum number of samplings was set as 1000, and the maximum number of clusters was set as 3. When we divided the image into 3x3 windows before applied the GLTS-QS algorithm, the algorithm could detect all clusters in 7.3 seconds, where the maximum number of samplings is set as small as 3.

For the image shown in Figure E.7 with coarse depth values, while the maximum number of samplings was set as 1000, the GLTS-QS algorithm identified all three clusters in 721.6 seconds that was much longer than the CPU time used for the same image with the floating values of depth. In addition, about 20% amount of points were taken as the outliers.

Through the experiments, we can see that the GLTS-QS algorithm is very time consuming for the range images with more clusters and more outliers. However, the window method can greatly improve its speed. From the above discussion, we also see

that the most computation load is from the multiple values of h and the greater number of samplings. If we can apply the FRHT method to obtain a coarse cluster, then use the GLTS-QS algorithm, namely, substitute the NFCQS process in the integrated clustering algorithm discussed in Chapter 5, we might need only one value of h, 0.5, and a smaller number of samplings for the GLTS-QS process. For the image shown in Figure E.7, we could detect all three clusters in 11.7 seconds when the FRHT with 3x3 windows was used before the GLTS-QS process with that one value of h, 0.5 was used and the maximum number of samplings was set as 1000, and about 7% amount of points were taken as the outliers. Obviously, the results indicate that the above integrated method works better than the GLTS-QS alone.

However, the above results are not as good as the results by the integrated clustering algorithm combining the FRHT and NFCQS, which was discussed in Chapter 5. The main reason is that, with the GLTS-QS algorithm, the final parameters of the true cluster are obtained from the values of nine points so they are not optimum. Actually, the FRHT has the same problem. Whereas for the NFCQS algorithm, the final results do not necessarily pass through any point but they are in sort of best match with all points. To improve the performance of GLTS-QS, the LS method and the fuzzy techniques might be employed to find the optimum results, which is the work at next step.

CHAPTER 9

CONCLUSIONS

9.1 Conclusions

There exist various clustering methods for pattern or object recognition. The focus of this dissertation was on the methods based on fuzzy techniques and Hough transform. Methods based on robust statistics similar to the minimum volume ellipsoid estimator were also considered. Based on the theoretical analyses of these methods conducted in this thesis, several novel clustering algorithms were proposed along with experimental testing of the methods. Although these new algorithms can be generally applied to both 2D and 3D images, the focus was more on the clustering techniques for 3D range images.

Based on the analyses of different HT methods, the techniques based on the randomized HT were found to be ideally suitable for images requiring a large number of parameters for the clusters. A variation of the RHT, called the fast randomized Hough transform was proposed. The key idea of FRHT is to divide the original image into multiple regions and apply random sampling method to map data points in the image space into the parameter space or feature space, then obtain the parameters of true clusters. This results in the following characteristics: high computation speed, low memory requirement, high result resolution and infinite parameter space. This method was shown to work well with many complex images, such as the quadric surface image containing many clusters and noisy points. Such performance is impossible for the conventional Hough transform

method. This method has also been successfully used in rather noisy 2D images such as the one shown in Figure 1.1.

Fuzzy clustering techniques are based on objective function optimization techniques and generate a fuzzy partition of the data. The Fuzzy c-Shells algorithm proposed by Dave [1990, 1991b] has proved to be a great success as a method of detecting and representing circular sub-structure in 2D data sets in digital images. A series of algorithms have been proposed using the "shell" concept [Dave et al. 1989, Dave and Patel 1990a, 1990b, Dave and Bhaswan 1992, Krishnapuram et al. 1992, etc.]. Krishnapuram et al. extended the circular shells to the general quadric shells and developed the Fuzzy C Quadric Shells clustering algorithm [Krishnapuram et al. 1993c]. Application of the FCQS clustering algorithm to 3D range data were discussed in this thesis along with experimental testing. It was shown that although with a very good membership initialization the FCQS can correctly cluster the data, its performance in noisy images with poor initialization is not very good.

Any valuable clustering algorithm should be able to deal with outliers or noisy points mixed in the data. Dave proposed a concept of "noise prototype" that could identify noisy data points as a separate cluster so the true clusters might be detected correctly [Dave 1991a]. Combining the concept of "noise prototype" with the regular fuzzy clustering algorithms created the noise type clustering algorithms, such as the Noise Fuzzy C Shells and Noise Fuzzy C Quadric Shells clustering algorithms. The author investigated these algorithms in detail, and showed through the experimental results that the noise type algorithms were robust against noise and might have faster convergence.

Whether or not the algorithm is a noise type, the major limitations are the need to know the number of clusters in advance and a need for a good membership initialization in order to achieve good clustering results. Whereas the FRHT method does not need such kind of prior information and even possesses the advantages of high detection speed and low storage requirement. But its results may not be very accurate as they are decided by the values of only a few points through solution of a set of parameter equations. In this dissertation, a novel integrated clustering algorithm combining the advantages of FRHT and NFCQS methods was proposed. It has proved to be a robust clustering algorithm through which the number of clusters is need not be determined in advance, the initial membership matrix is not required, the detection accuracy is greatly improved and the computation speed is very fast. The development of this algorithm is one of the major contributions of this work.

The Least Squares regression method and the Least Median of Squares method are also analyzed in this work. The LS method has a theoretical breakdown point of 0% and its results are easily influenced, whereas the LMS has a theoretical breakdown point of 50% and its results are robust. Unfortunately, even LMS can not work well for the data with multiple clusters in absence of presegmentation. The Generalized Minimum Volume Ellipsoid estimator is a robust clustering method that can be used solve many pattern recognition problems in computer vision [Jolion et al. 1991]. Based on the idea of GMVE, the generalized Least Trimmed Squares for Quadric Surfaces (GLTS-QS) algorithm was developed. It is a robust clustering algorithm but has very slow computation speed because of extensive computations. By using the similar window method as in the FRHT,

its speed can increase greatly. If the FRHT is applied before the GLTS-QS, then its speed might further increase. In other words, the clustering method combining the FRHT and the GLTS-QS can improve clustering performance.

According to the optimality of results, the clustering algorithms discussed here can be classified into two categories: optimized clustering algorithm and non-optimized algorithm. The optimized clustering algorithms include all types of fuzzy c shells algorithms and LS method, where the final parameters are in sort of best match with all data points. The non-optimized algorithms include FRHT and LMS methods, where the final parameters are obtained from the values of a few points so they are not optimum. Therefore, in some situations, combining two types of algorithms is a good idea, as demonstrated by the integrated algorithm that combines FRHT and NFCQS methods.

From the procedural point of view, the clustering algorithms can be classified into two types. The one is that all clusters are detected simultaneously, such as in fuzzy clustering algorithms. The other is that all clusters are identified one by one, i.e. clusters are found one at a time, such as the FRHT method and the GLTS method.

In all these range image clustering algorithms, the computation of distance of a point from a cluster boundary is a must. Usually, there are three kinds of distance, the Euclidean distance or geometric distance, the approximate distance and the algebraic distance. The Euclidean distance is the exact distance that can provide reliable results but has the slowest computation speed. The approximate distance is the first-order approximation of the exact distance that has very fast computation speed and can achieve reliable results in most situations but is sort of nonlinear distance. The algebraic distance is computed

directly from the model equation, has the fastest computation speed and can work well in some situations but is highly nonlinear distance. The analyses and experimental results indicated that the approximate distance is preferred in most cases due to its features of high speed and reliability.

Application of the robust clustering methods to the granular flow research project was considered. We developed the D&C-NFCS algorithm to detect the exact locations of spherical particles from an extremely noisy image, where the regular HT was first used to obtain the coarse results, then the NFCS method was used to refine the results. The experimental results showed that the D&C-NFCS algorithm is an efficient clustering method for noisy images.

Clustering validity is an important issue in clustering techniques. In this dissertation, recent developments in cluster validity for shell clusters were reviewed, and a new validity criterion for circular clusters was proposed. The new validity method was developed by considering the distribution of the points on the circular edge, and its performance was tested through examples.

9.2 Future Work

In this dissertation, different clustering methods for both intensity images and ranges images are investigated, and were shown to be effective against noisy data, computationally fast, and did not depend on initializations. Nevertheless, the work presented here can be only considered as a start of more extensive studies based on the ideas presented. Extensive investigation and further research are needed in order to make

these techniques more reliable and applicable to the problems in manufacturing industry and other fields.

The algorithms proposed here are used for the general quadratic shapes, but in some practical applications, the clusters may be some specific simple shapes, for example, in the machine manufacturing industry, many parts are of cylindrical shape. The algorithm developed for a particular case will have faster speed and can achieve more reliable results. The clustering approach can be coupled with learning schemes to make it an automatic recognition system.

As mentioned before, the GMVE is a robust clustering algorithm, but needs a great deal of computations. To make the methods based on it more efficient, its computation load needs to be reduced. Those ideas and methods used for reducing computation in HT might be applied to increase its speed. For example, the experiments showed that the window based method was able to reduce computation load for GLTS-QS. However, an extensive investigation is needed for the GMVE.

In order for the approach proposed here, namely, finding one cluster at a time, to work, one must know how to test for a *good* cluster. Reliable cluster validity criteria are a must for this technique as they can be used not only to measure the goodness of the final results, but also to enhance some clustering algorithms, for example, in the GLTS-QS algorithm, an effective clustering validity criterion for measuring the validity of all possible clusters at different values of h would definitely improve the clustering performance. Ideas for circular clusters are proposed here but more work is required for developing a general approach.

APPENDIX A

DSCRIPTION OF SUBROUTINE HYBRD IN MINPACK

```
/*    subroutine hybrd */

/*    the purpose of hybrd is to find a zero of a system of */
/*    n nonlinear functions in n variables by a modification */
/*    of the powell hybrid method. the user must provide a */
/*    subroutine which calculates the functions. the jacobian is */
/*    then calculated by a forward-difference approximation. */

/*    the subroutine statement is */

/*      subroutine hybrd(fcn,n,x,fvec,xtol,maxfev,ml,mu,epsfcn, */
/*                       diag,mode,factor,nprint,info,nfev,fjac, */
/*                       ldfjac,r,lr,qtf,wa1,wa2,wa3,wa4) */

/*    where */

/*      fcn is the name of the user-supplied subroutine which */
/*        calculates the functions. fcn must be declared */
/*        in an external statement in the user calling */
/*        program, and should be written as follows. */

/*        subroutine fcn(n,x,fvec,iflag) */
/*        integer n,iflag */
/*        double precision x(n),fvec(n) */
/*        ---------- */
/*        calculate the functions at x and */
/*        return this vector in fvec. */
/*        --------- */
/*        return */
/*        end */

/*        the value of iflag should not be changed by fcn unless */
/*        the user wants to terminate execution of hybrd. */
/*        in this case set iflag to a negative integer. */

/*      n is a positive integer input variable set to the number */
/*        of functions and variables. */

/*      x is an array of length n. on input x must contain */
```

```
/*      an initial estimate of the solution vector. on output x */
/*      contains the final estimate of the solution vector. */

/*    fvec is an output array of length n which contains */
/*      the functions evaluated at the output x. */

/*    xtol is a nonnegative input variable. termination */
/*      occurs when the relative error between two consecutive */
/*      iterates is at most xtol. */

/*    maxfev is a positive integer input variable. termination */
/*      occurs when the number of calls to fcn is at least maxfev */
/*      by the end of an iteration. */

/*    ml is a nonnegative integer input variable which specifies */
/*      the number of subdiagonals within the band of the */
/*      jacobian matrix. if the jacobian is not banded, set */
/*      ml to at least n - 1. */

/*    mu is a nonnegative integer input variable which specifies */
/*      the number of superdiagonals within the band of the */
/*      jacobian matrix. if the jacobian is not banded, set */
/*      mu to at least n - 1. */

/*    epsfcn is an input variable used in determining a suitable */
/*      step length for the forward-difference approximation. this */
/*      approximation assumes that the relative errors in the */
/*      functions are of the order of epsfcn. if epsfcn is less */
/*      than the machine precision, it is assumed that the relative */
/*      errors in the functions are of the order of the machine */
/*      precision. */

/*    diag is an array of length n. if mode = 1 (see */
/*      below), diag is internally set. if mode = 2, diag */
/*      must contain positive entries that serve as */
/*      multiplicative scale factors for the variables. */

/*    mode is an integer input variable. if mode = 1, the */
/*      variables will be scaled internally. if mode = 2, */
/*      the scaling is specified by the input diag. other */
/*      values of mode are equivalent to mode = 1. */

/*    factor is a positive input variable used in determining the */
/*      initial step bound. this bound is set to the product of */
/*      factor and the euclidean norm of diag*x if nonzero, or else */
```

```
/*      to factor itself. in most cases factor should lie in the */
/*      interval (.1,100.). 100. is a generally recommended value. */

/*    nprint is an integer input variable that enables controlled */
/*      printing of iterates if it is positive. in this case, */
/*      fcn is called with iflag = 0 at the beginning of the first */
/*      iteration and every nprint iterations thereafter and */
/*      immediately prior to return, with x and fvec available */
/*      for printing. if nprint is not positive, no special calls */
/*      of fcn with iflag = 0 are made. */

/*    info is an integer output variable. if the user has */
/*      terminated execution, info is set to the (negative) */
/*      value of iflag. see description of fcn. otherwise, */
/*      info is set as follows. */

/*      info = 0   improper input parameters. */

/*      info = 1   relative error between two consecutive iterates */
/*                 is at most xtol. */

/*      info = 2   number of calls to fcn has reached or exceeded */
/*                 maxfev. */

/*      info = 3   xtol is too small. no further improvement in */
/*                 the approximate solution x is possible. */

/*      info = 4   iteration is not making good progress, as */
/*                 measured by the improvement from the last */
/*                 five jacobian evaluations. */

/*      info = 5   iteration is not making good progress, as */
/*                 measured by the improvement from the last */
/*                 ten iterations. */

/*    nfev is an integer output variable set to the number of */
/*      calls to fcn. */

/*    fjac is an output n by n array which contains the */
/*      orthogonal matrix q produced by the qr factorization */
/*      of the final approximate jacobian. */

/*    ldfjac is a positive integer input variable not less than n */
/*      which specifies the leading dimension of the array fjac. */
```

```
/*     r is an output array of length lr which contains the */
/*       upper triangular matrix produced by the qr factorization */
/*       of the final approximate jacobian, stored rowwise. */

/*     lr is a positive integer input variable not less than */
/*       (n*(n+1))/2. */

/*     qtf is an output array of length n which contains */
/*       the vector (q transpose)*fvec. */

/*     wa1, wa2, wa3, and wa4 are work arrays of length n. */

/*   subprograms called */

/*     user-supplied ..... fcn */

/*     minpack-supplied ... dogleg,dpmpar,enorm,fdjac1, */
/*                          qform,qrfac,r1mpyq,r1updt */

/*     fortran-supplied ... dabs,dmax1,dmin1,min0,mod */

/*   argonne national laboratory. minpack project. march 1980. */
/*   burton s. garbow, kenneth e. hillstrom, jorge j. more */
```

APPENDIX B

A TABLE ABOUT NUMBER OF RANDOM SUBSAMPLES IN LMS

Table B.1 Number m of Random Subsamples, Determined in Function of p and ε by Requiring That the Probability of at Least One Good Subsample Is 95% or More [Rousseeuw *et al.* 1987].

Dimension	Fraction ε of Contaminated Data						
p	5%	10%	20%	25%	30%	40%	50%
1	1	2	2	3	3	4	5
2	2	2	3	4	5	7	11
3	2	3	5	6	8	13	23
4	2	3	6	8	11	22	47
5	3	4	8	12	17	38	95
6	3	4	10	16	24	63	191
7	3	5	13	21	35	106	382
8	3	6	17	29	51	177	766
9	4	7	21	36	73	296	1533
10	4	7	27	52	105	494	3067

APPENDIX C

ADDITIONAL FIGURES FOR CHAPTER 2

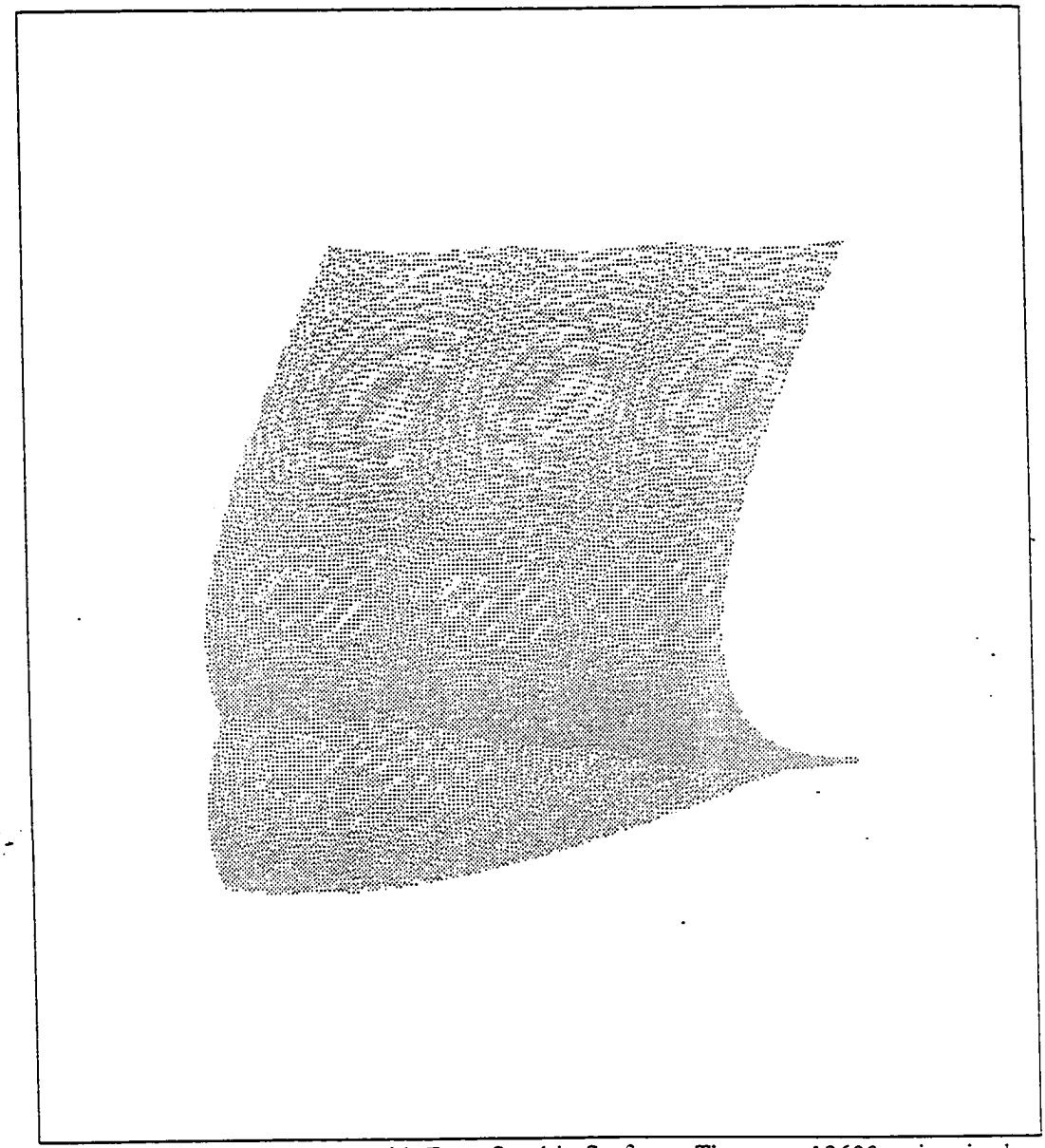


Figure C.1 A Range Image with Four Quadric Surfaces. There are 19600 points in the image and the parameters of all four clusters are shown as Table 2.13.

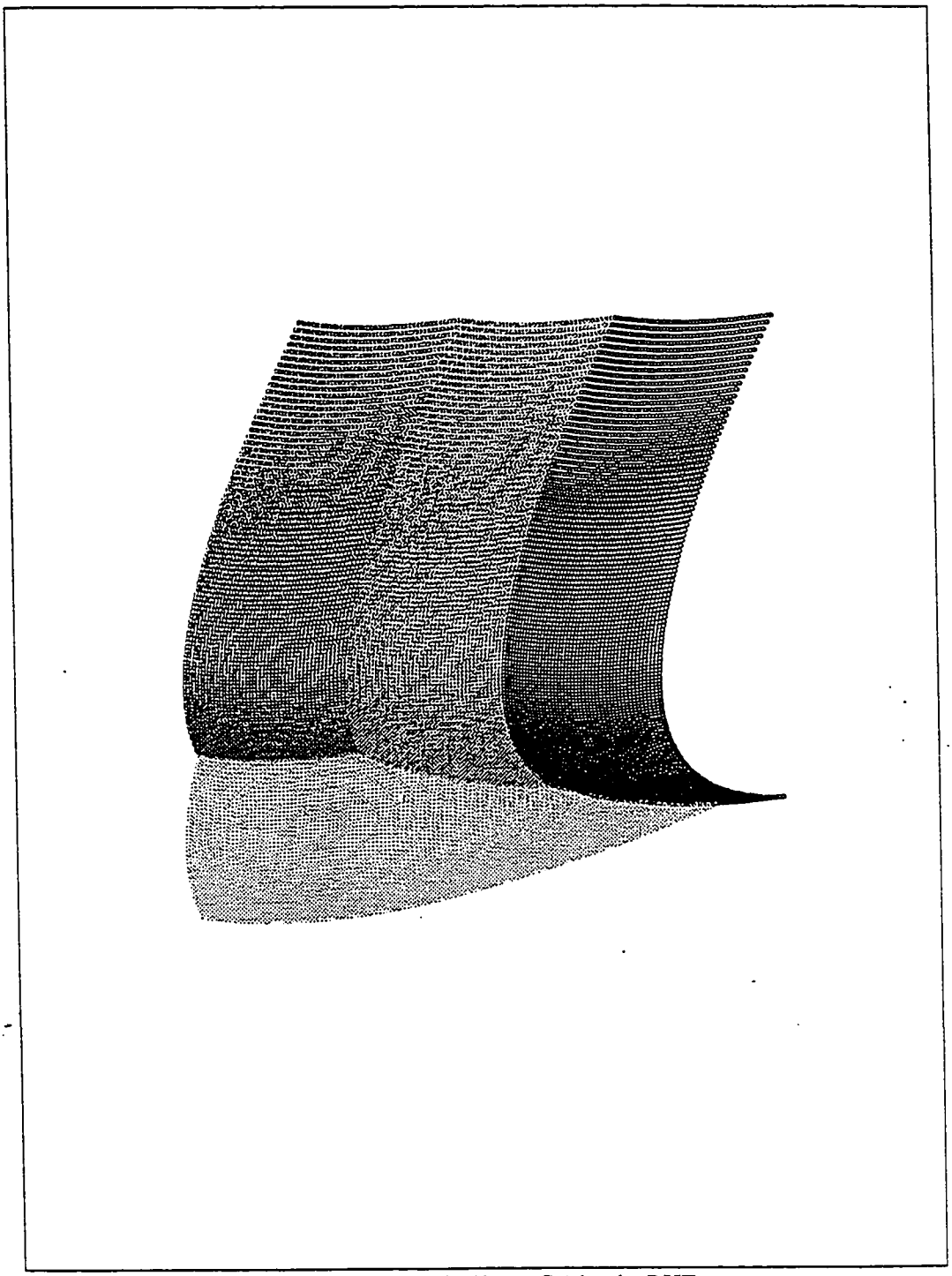


Figure C.2 Results of the Image Shown in Figure C.1 by the RHT.

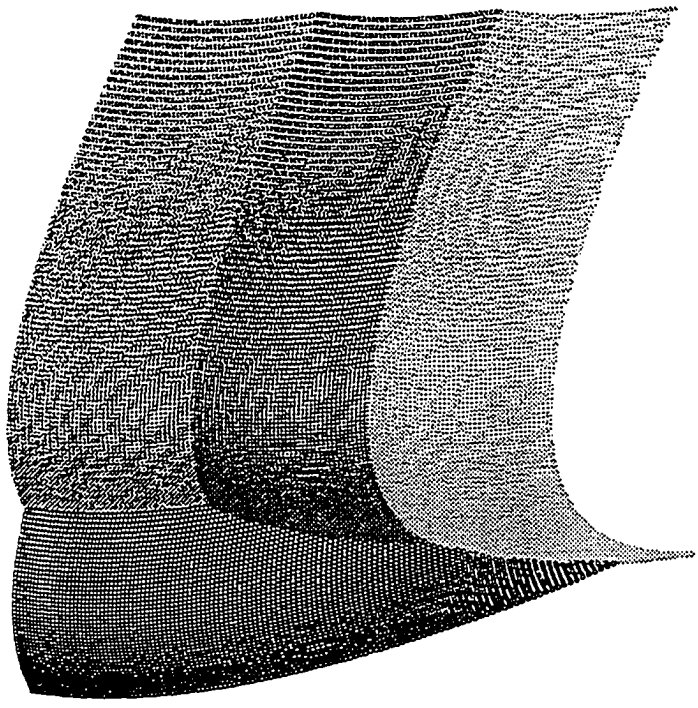


Figure C.3 Results of the Image Shown in Figure C.1 by the FRHT with 3x2 Regions.

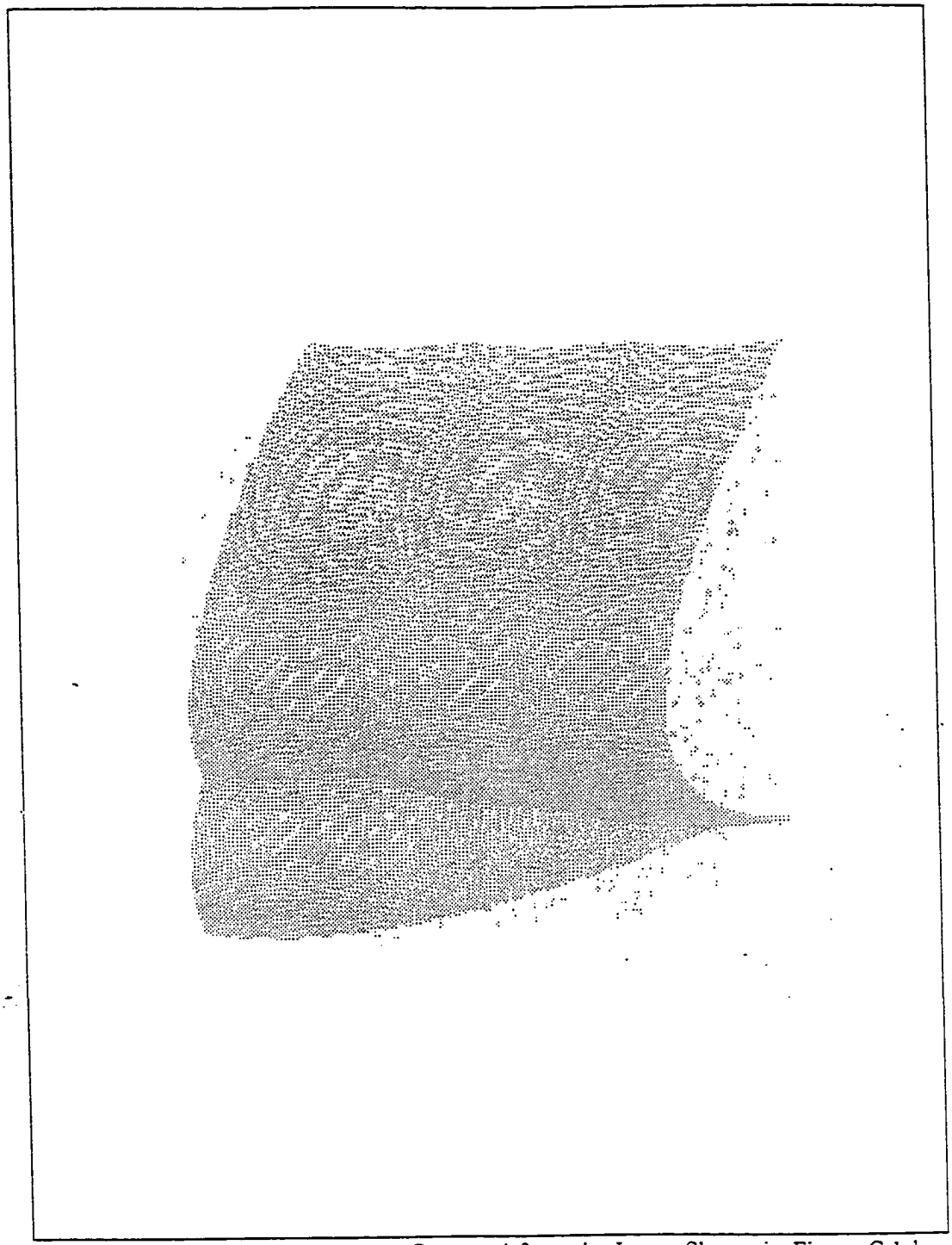


Figure C.4 A Noise Range Image Generated from the Image Shown in Figure C.1 by Adding 10% Noise Points.

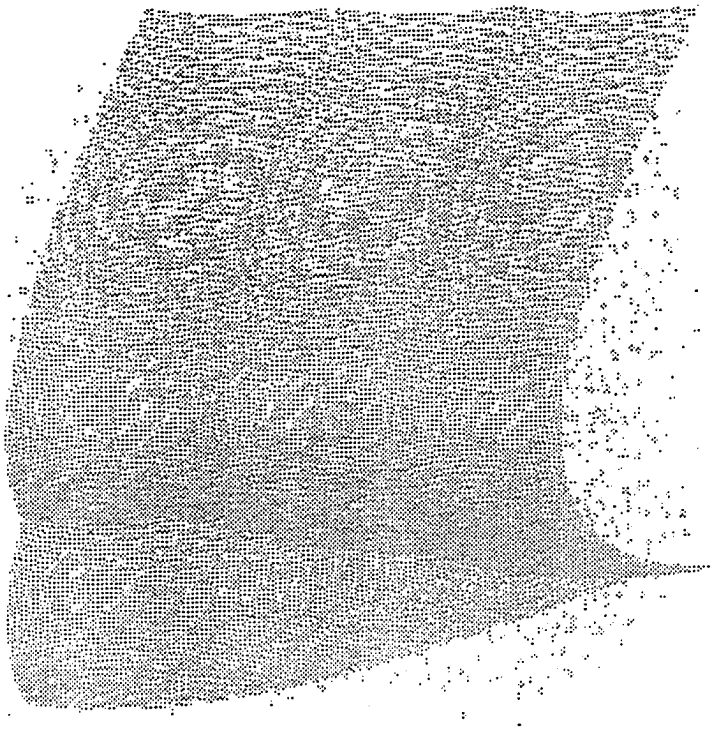


Figure C.5 A Noise Range Image Generated from the Image Shown in Figure C.1 by Adding 20% Noise Points.

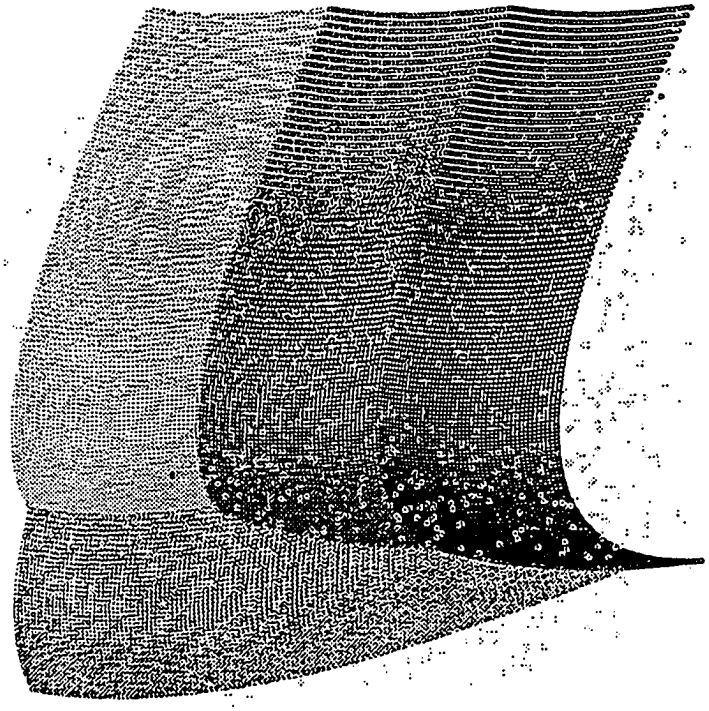


Figure C.6 Results of the Image Shown in Figure C.4 by the RHT.

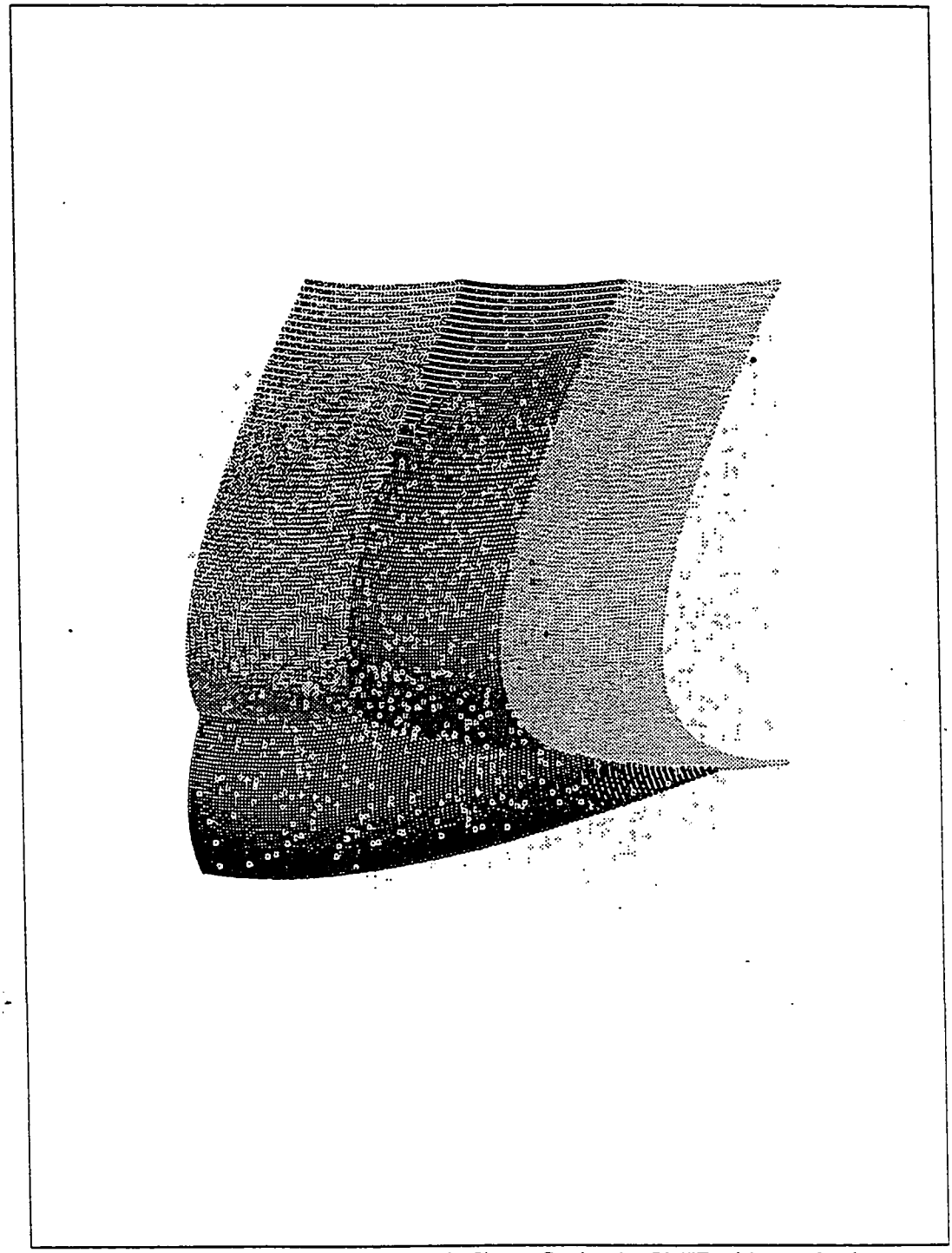


Figure C.7 Results of the Image Shown in Figure C.4 by the FRHT with 3x2 Regions.

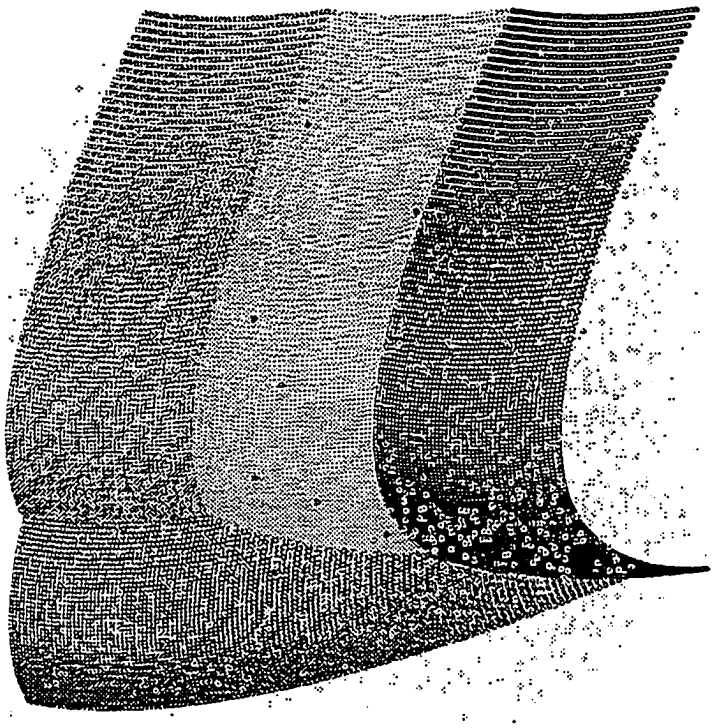


Figure C.8 Results of the Image Shown in Figure C.5 by the RHT.

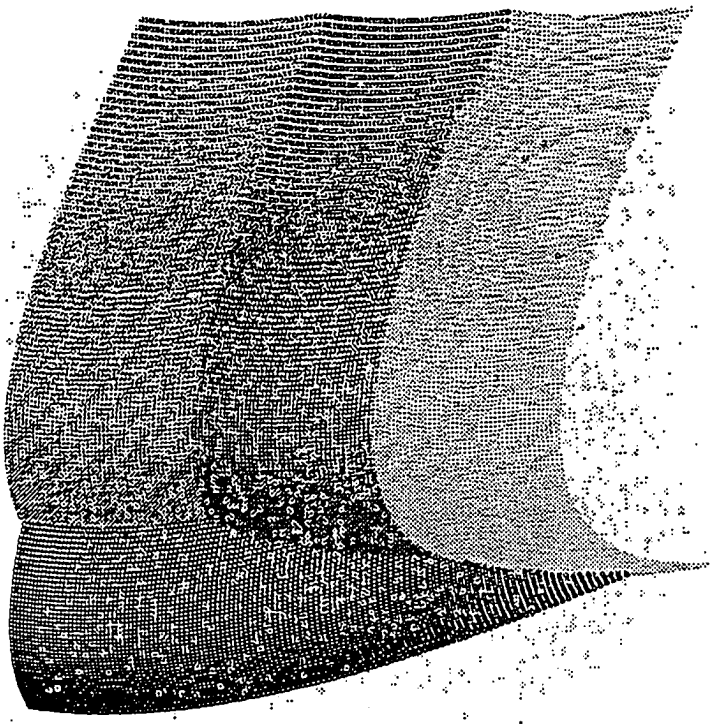


Figure C.9 Results of the Image Shown in Figure C.5 by the FRHT with 3x2 Regions.

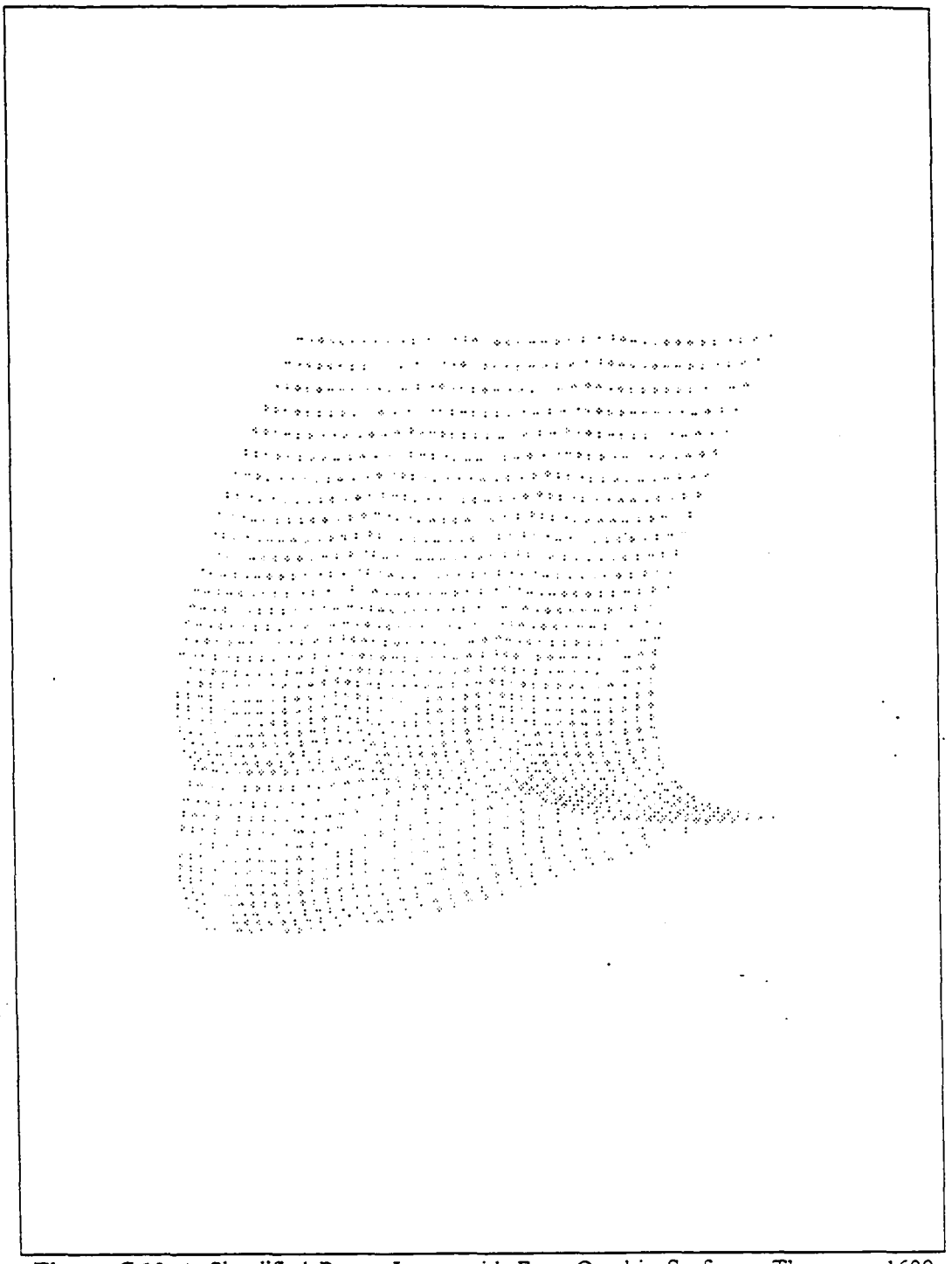


Figure C.10 A Simplified Range Image with Four Quadric Surfaces. There are 1600 points in the image and the parameters of all clusters are shown in Table 2.13.

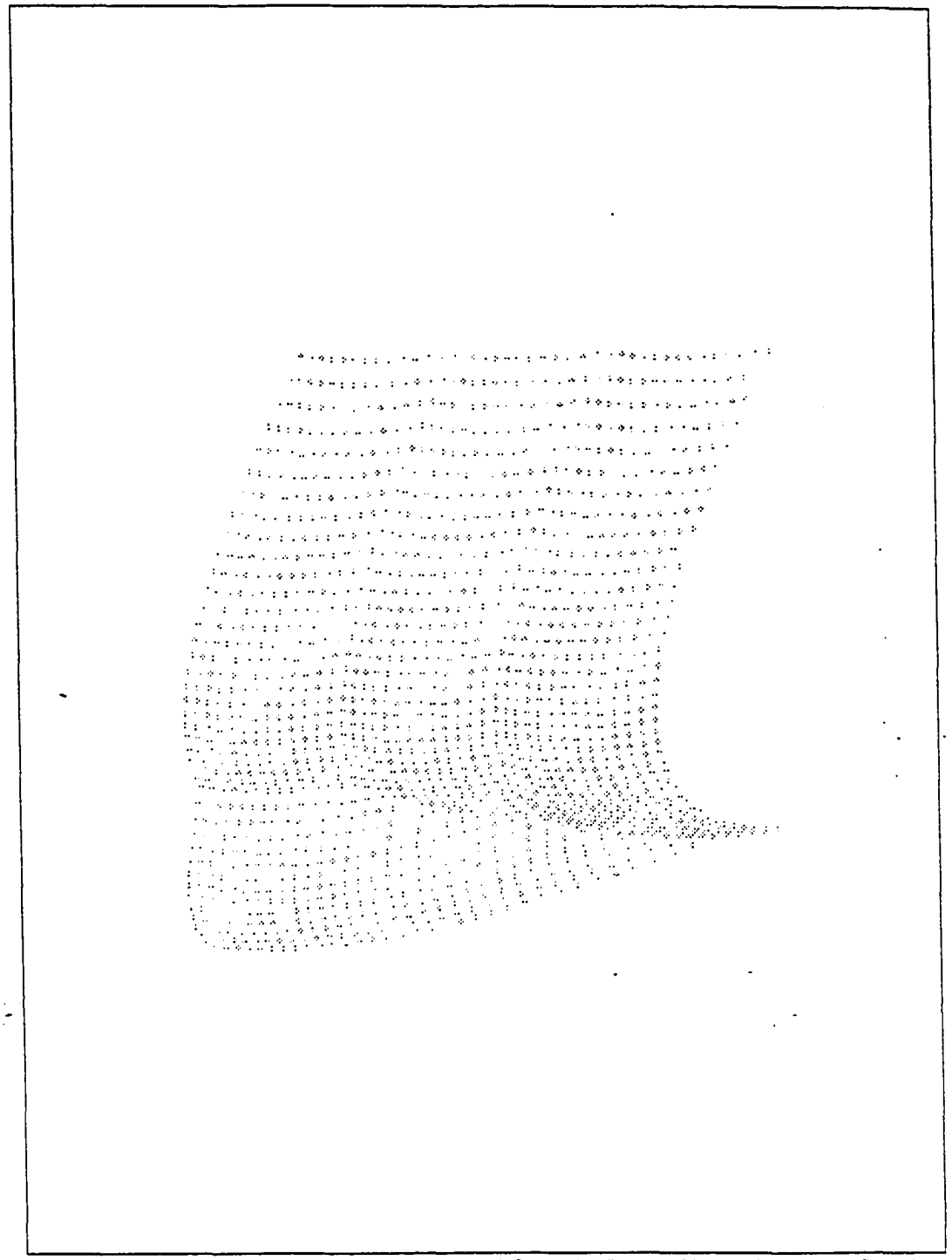


Figure C.11 A Noise Range Image Generated from the Image Shown in Figure C.10 by Randomly Shifting all Points Along the Depth Direction.

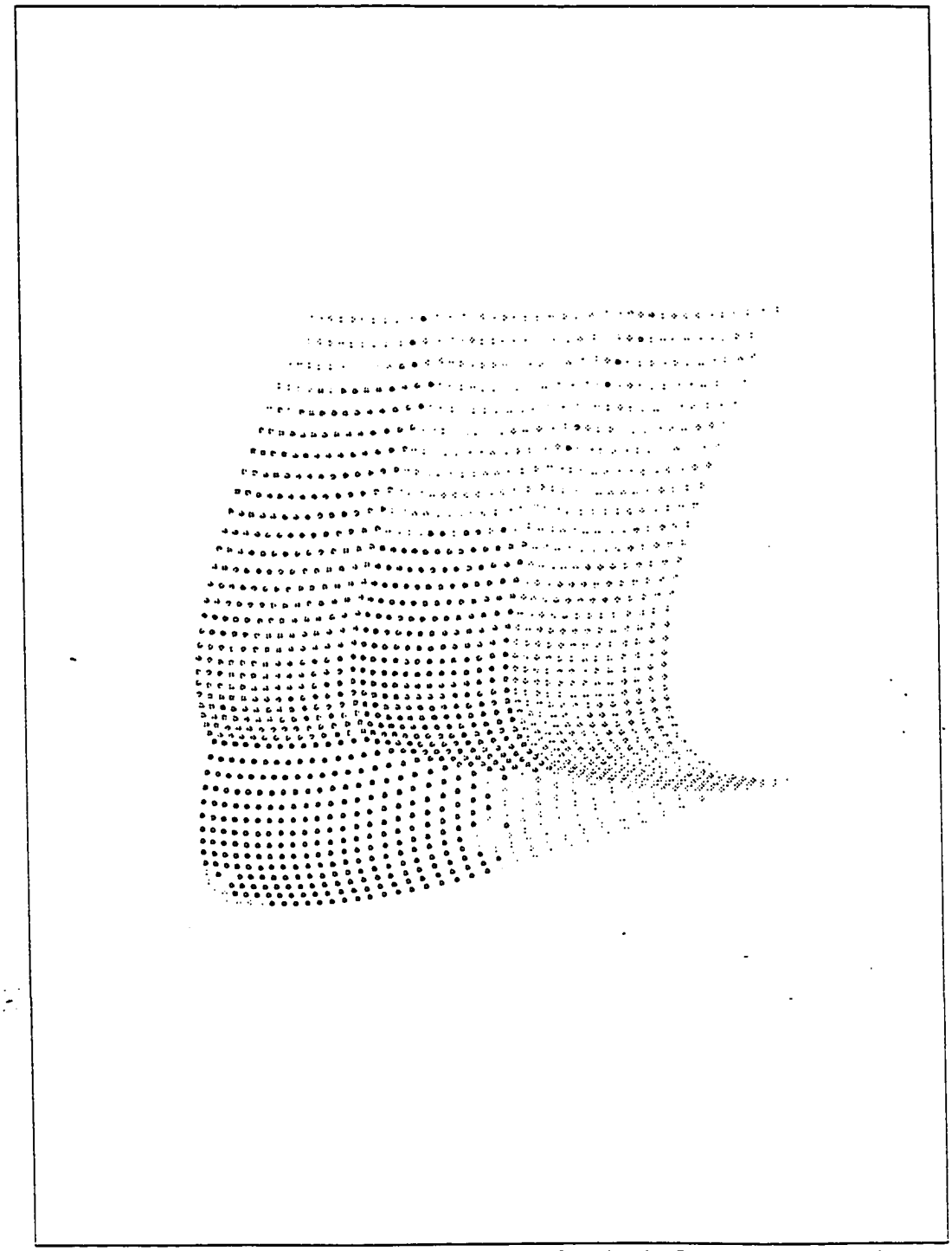


Figure C.12 Results of the Image Shown in Figure C.11 by the FRHT with 3x2 Regions.

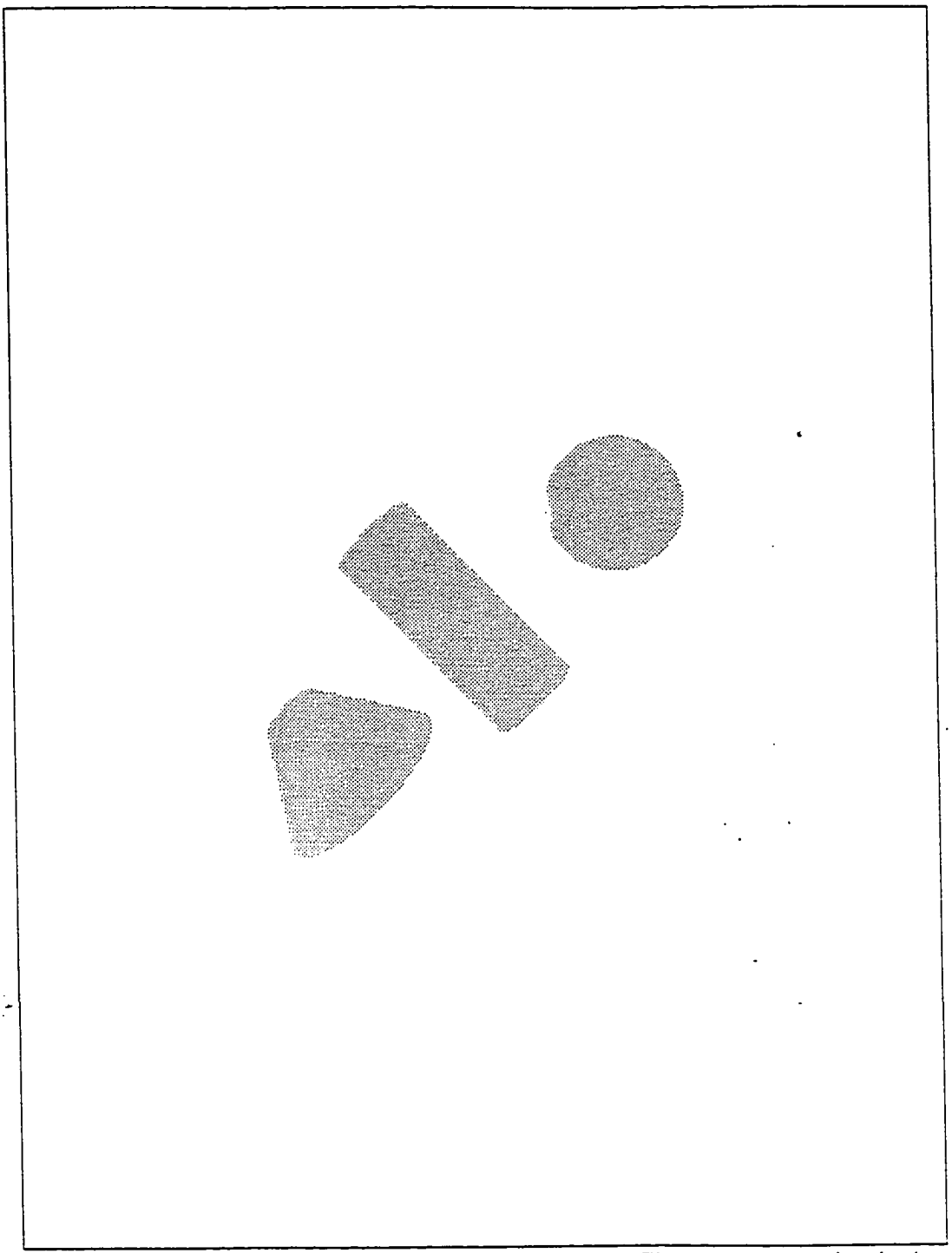


Figure C.13 A Range Image with Three Quadric Shapes. There are 4062 points in the image and the depth values are recorded in floating numbers.

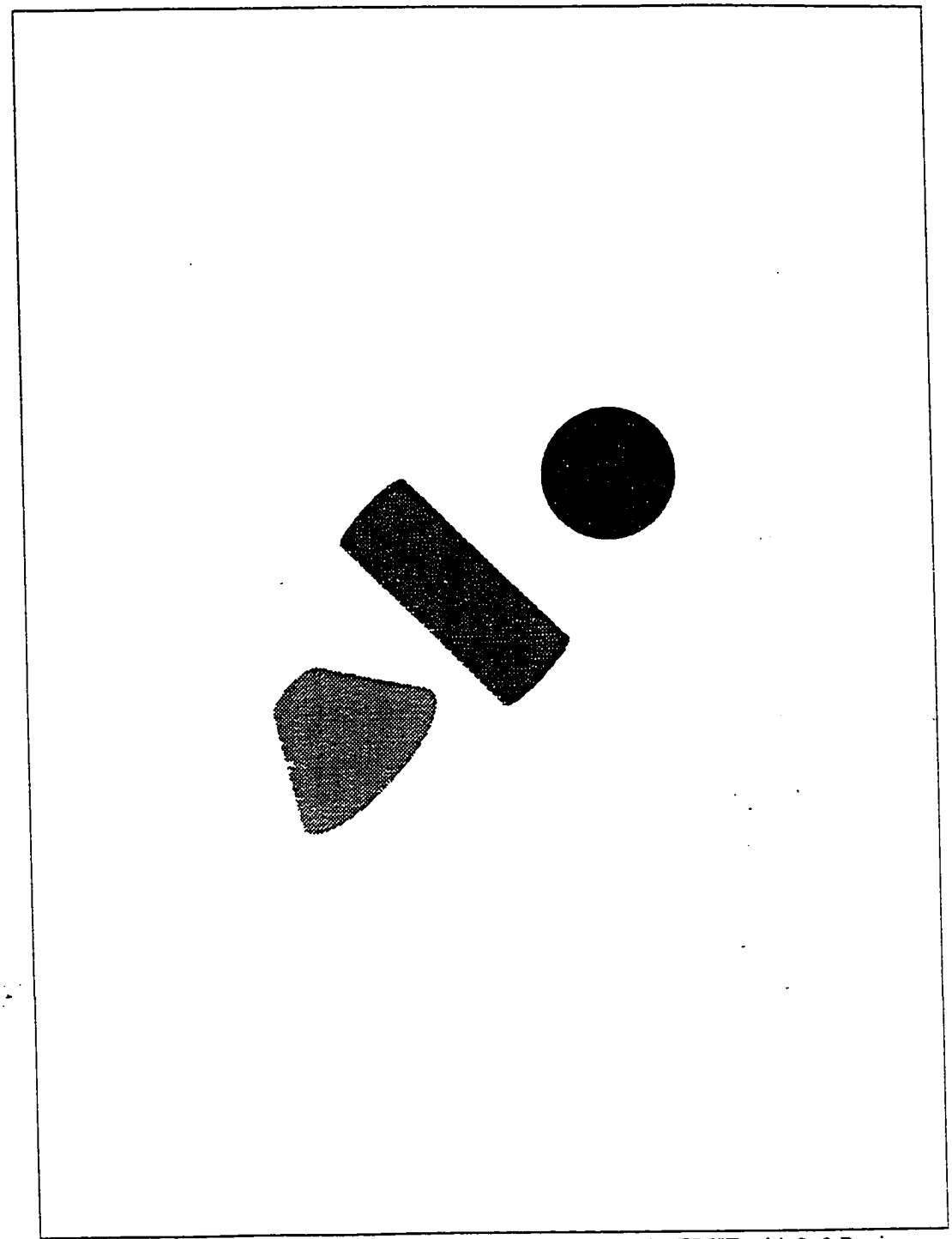


Figure C.14 Results of the Image Shown in Figure C.13 by the FRHT with 3x3 Regions.

APPENDIX D

FIGURES FOR CHAPTER 4

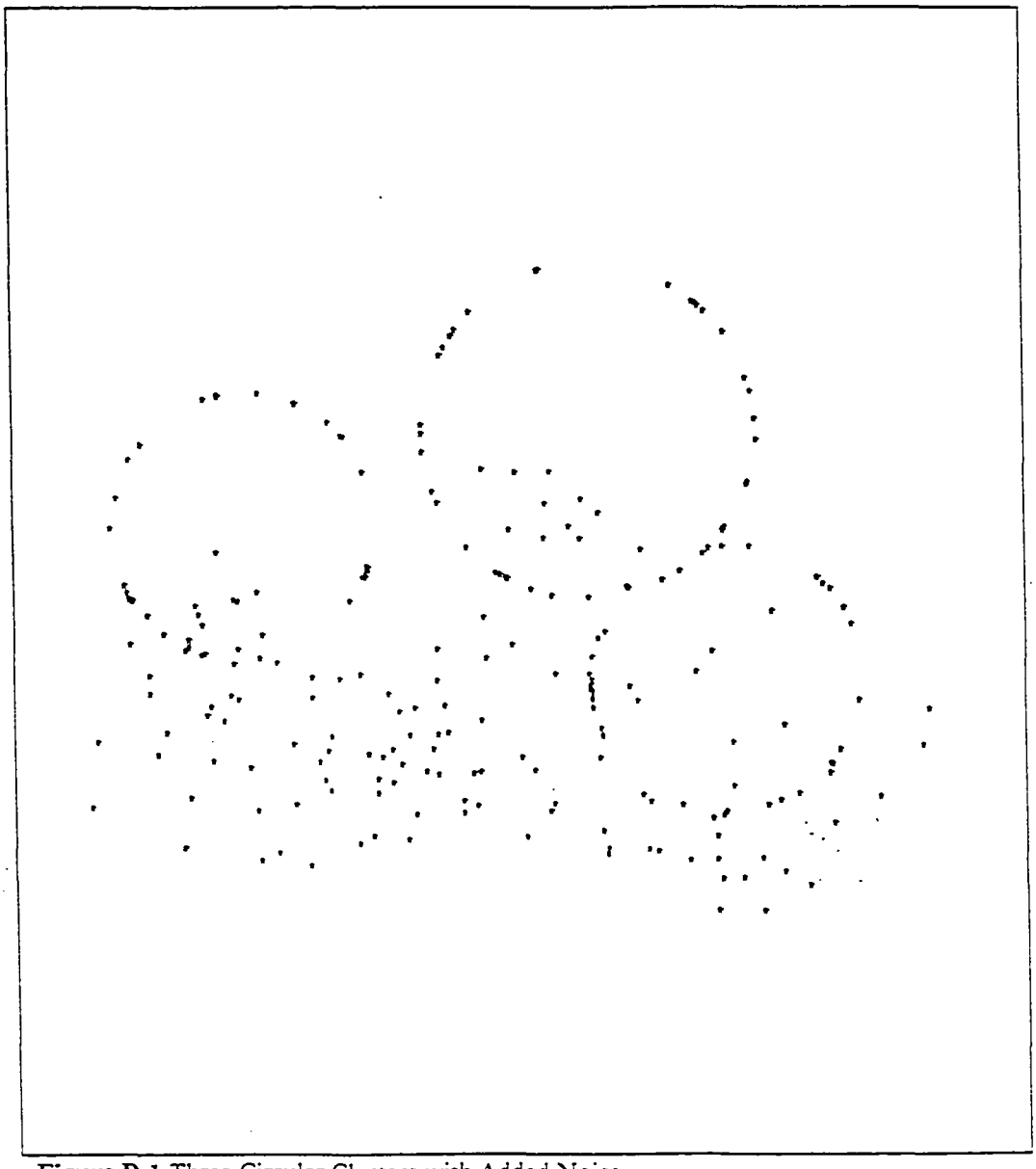


Figure D.1 Three Circular Clusters with Added Noise.

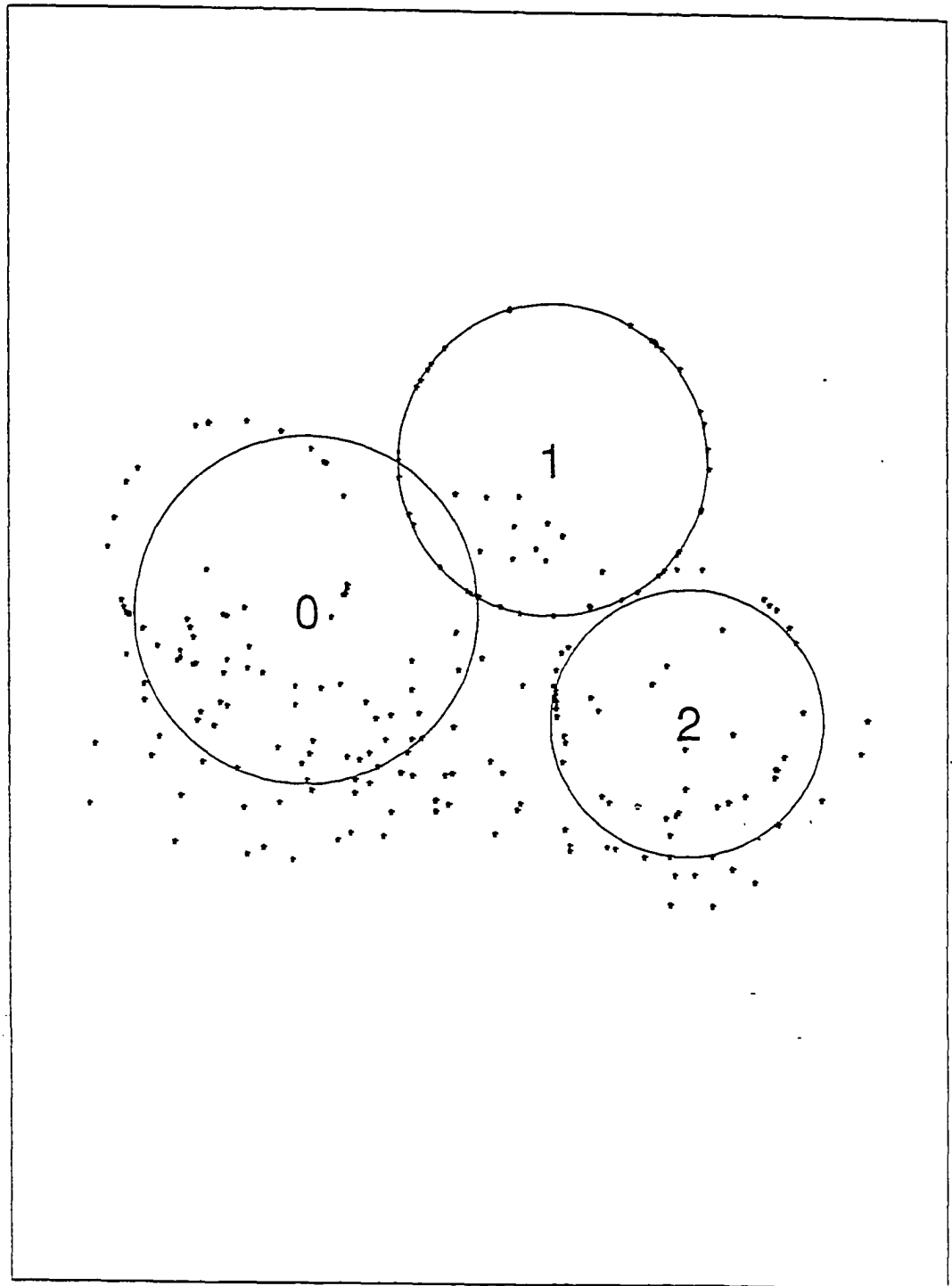


Figure D.2 Results of the Image Shown in Figure D.1 by the Conventional FCS Algorithm.

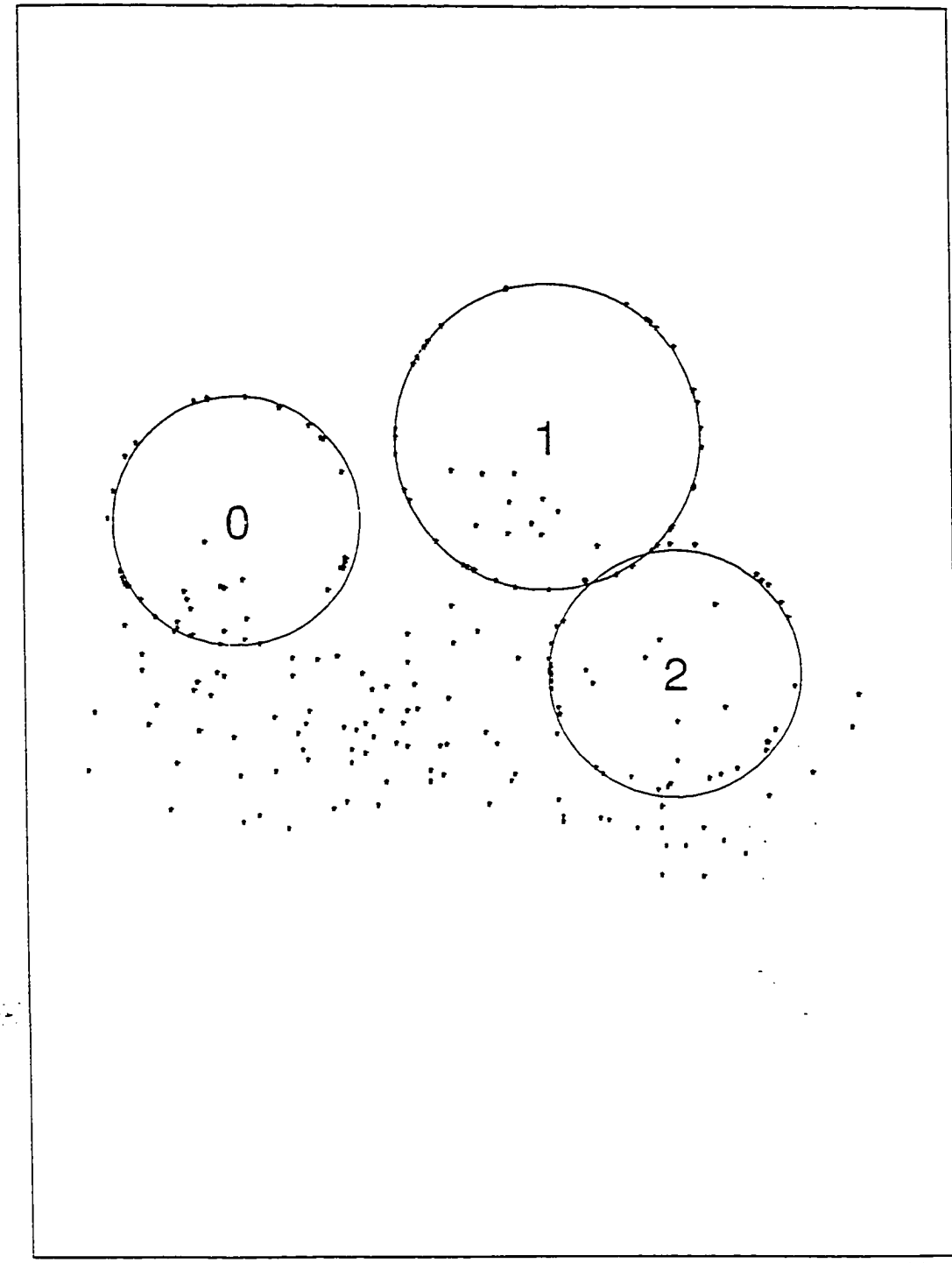


Figure D.3 Results of the Image Shown in Figure D.1 by the Noise Clustering Algorithm (NFCS).

APPENDIX E

FIGURES FOR CHAPTER 5

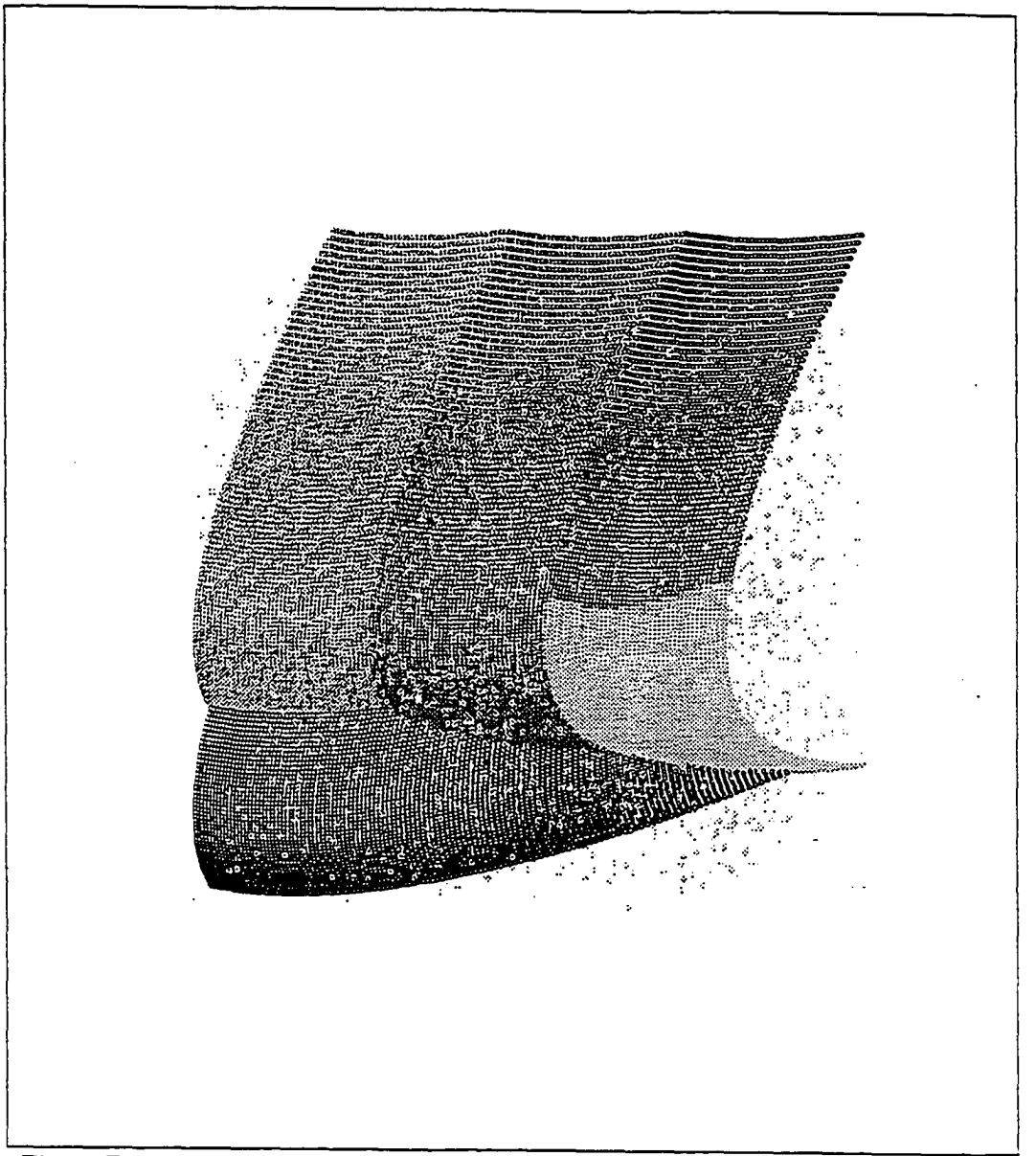


Figure E.1 Results of the Image Shown in Figure C.5 by the FRHT with 3x3 Regions.

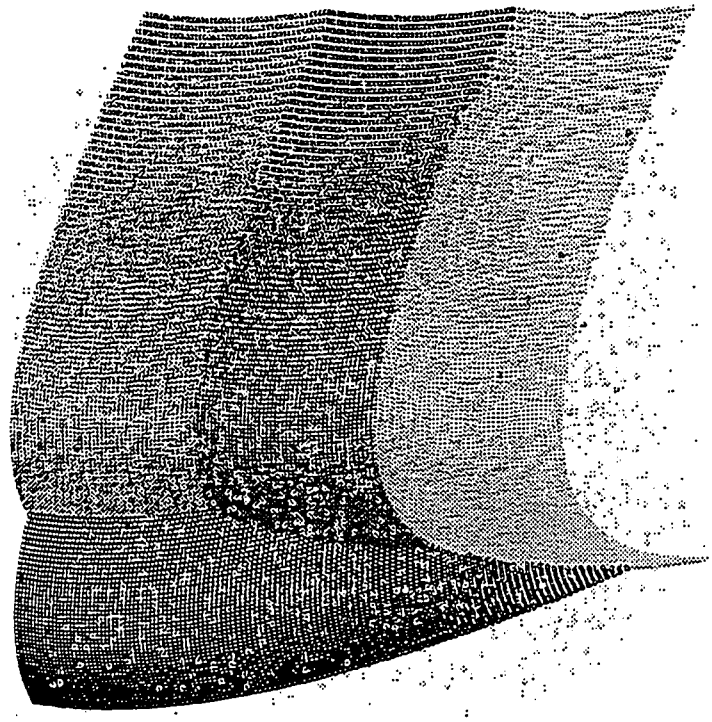


Figure E.2 Results of the Image Shown in Figure C.5 by the Integrated Algorithm.

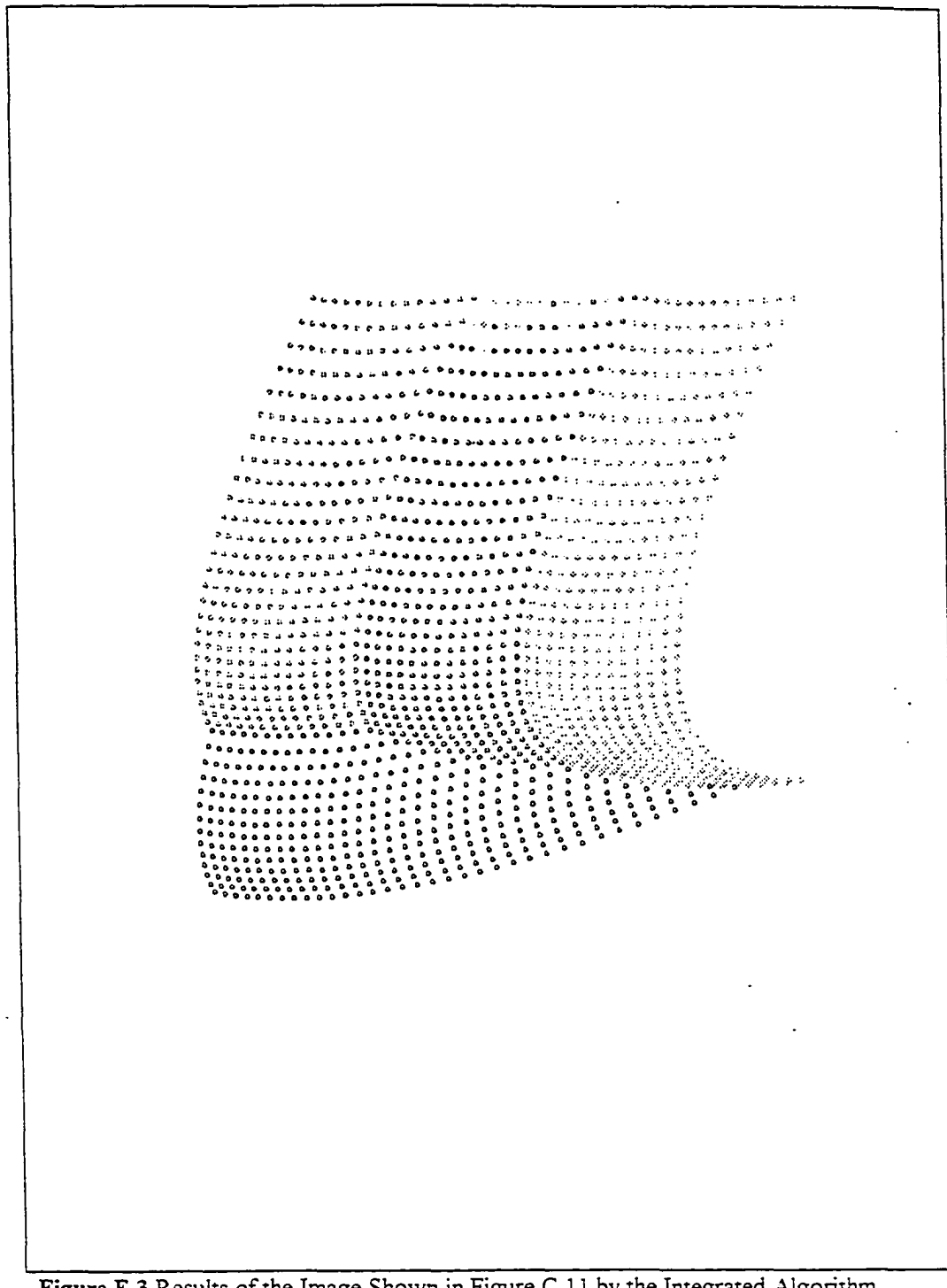


Figure E.3 Results of the Image Shown in Figure C.11 by the Integrated Algorithm.

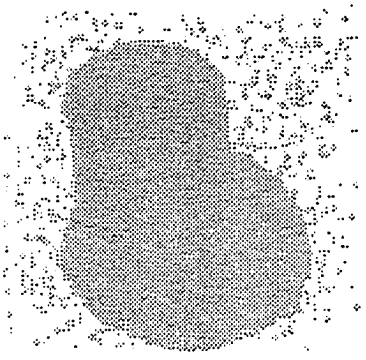


Figure E.4 A Noise Range Image Generated from Two Real Images by Adding Some Noise Points.

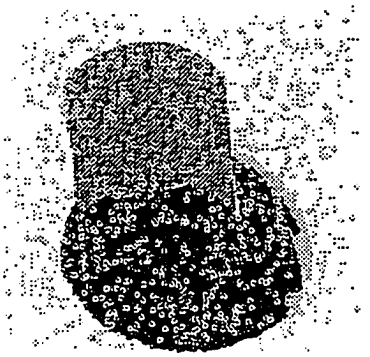


Figure E.5 Results of the Image Shown in Figure E.4 by the FRHT with 1x2 Regions.

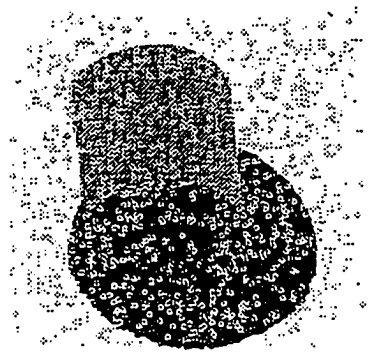


Figure E.6 Results of the Image Shown in Figure E.4 by the Integrated Algorithm.

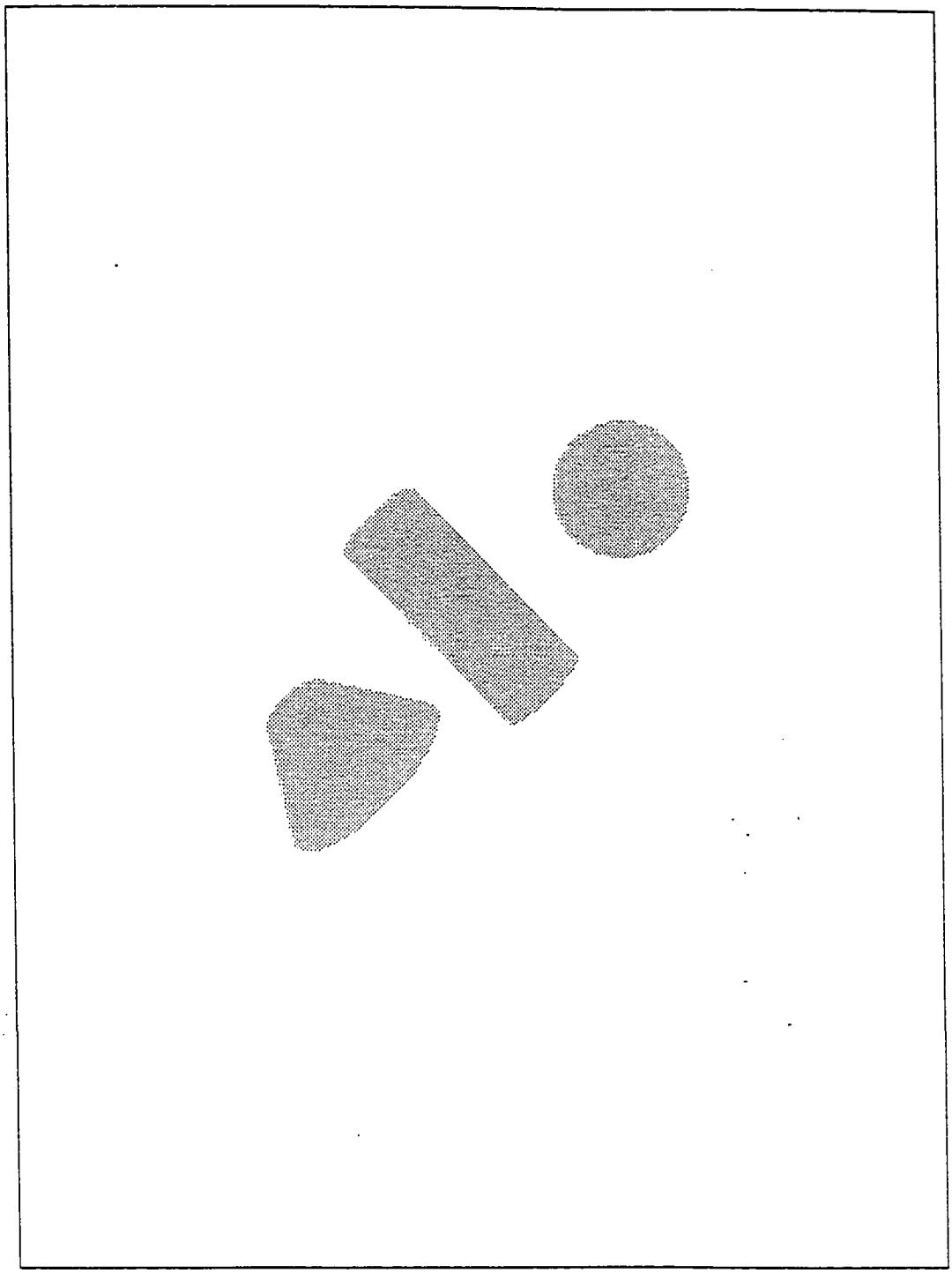


Figure E.7 A Noise Range Image with Three Quadric Shapes. There are 4062 points in the image and the depth values are recorded in integer numbers.

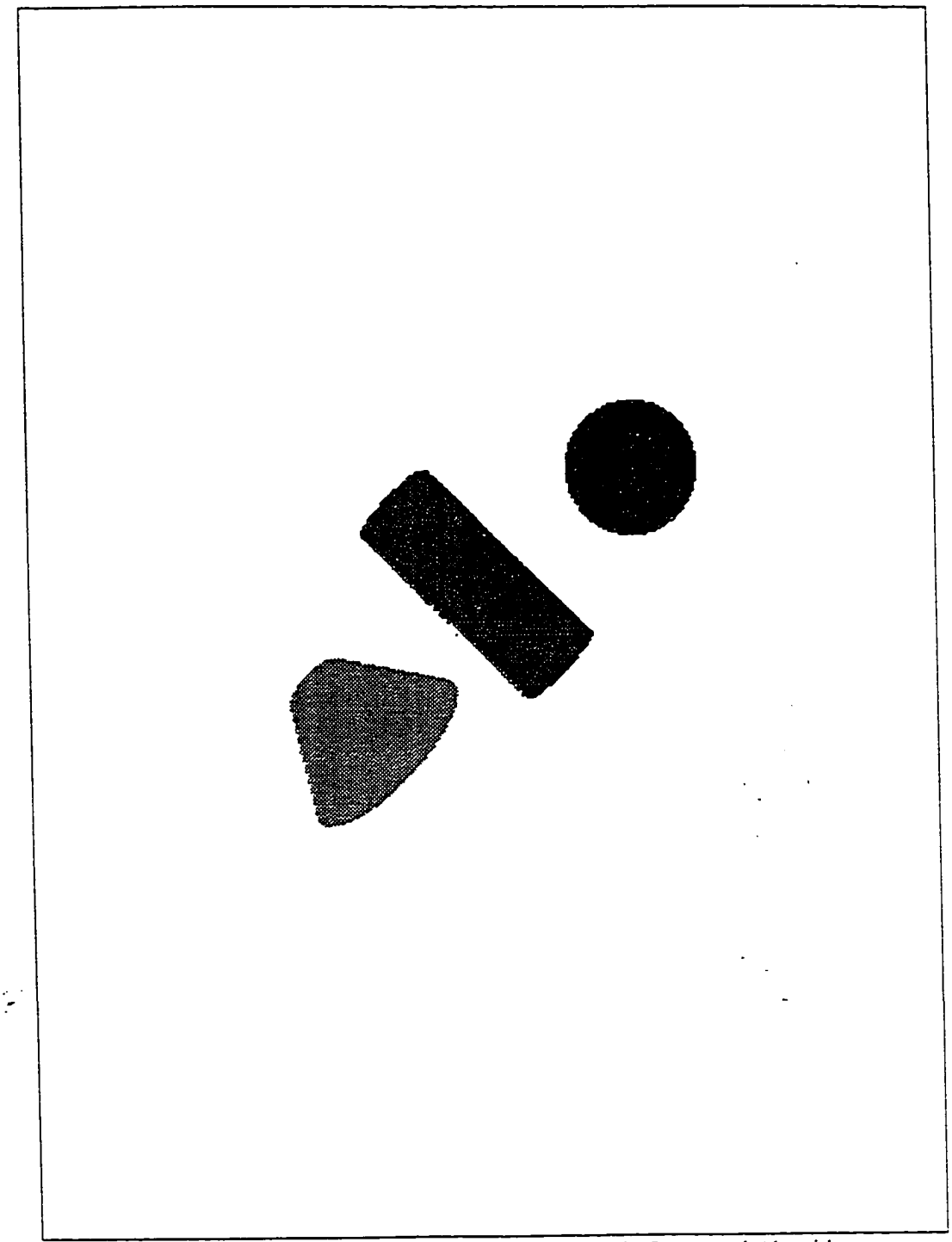


Figure E.8 Results of the Image Shown in Figure E.7 by the Integrated Algorithm.

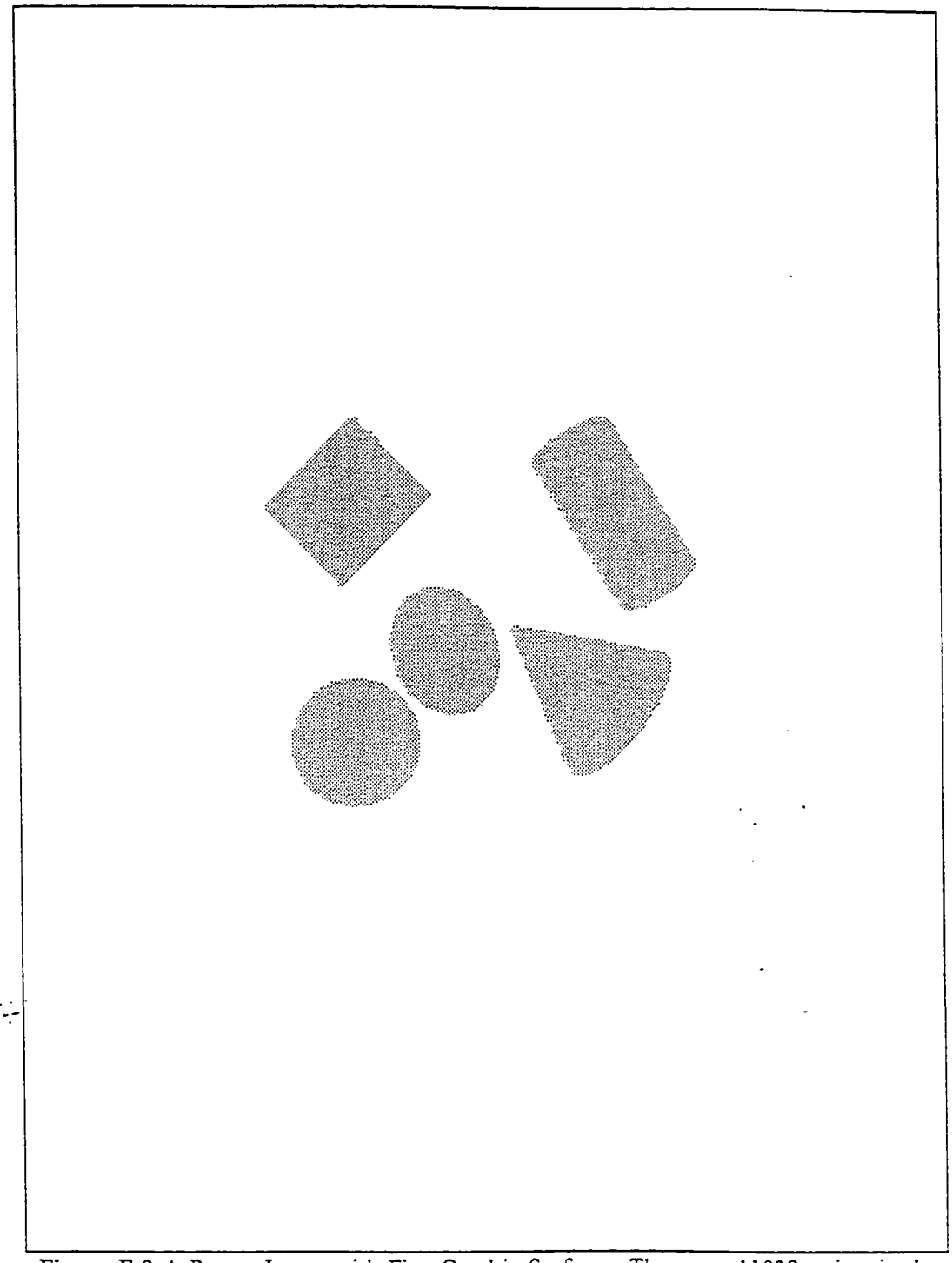


Figure E.9 A Range Image with Five Quadric Surfaces. There are 11038 points in the image and the depth values are recorded in integer numbers.

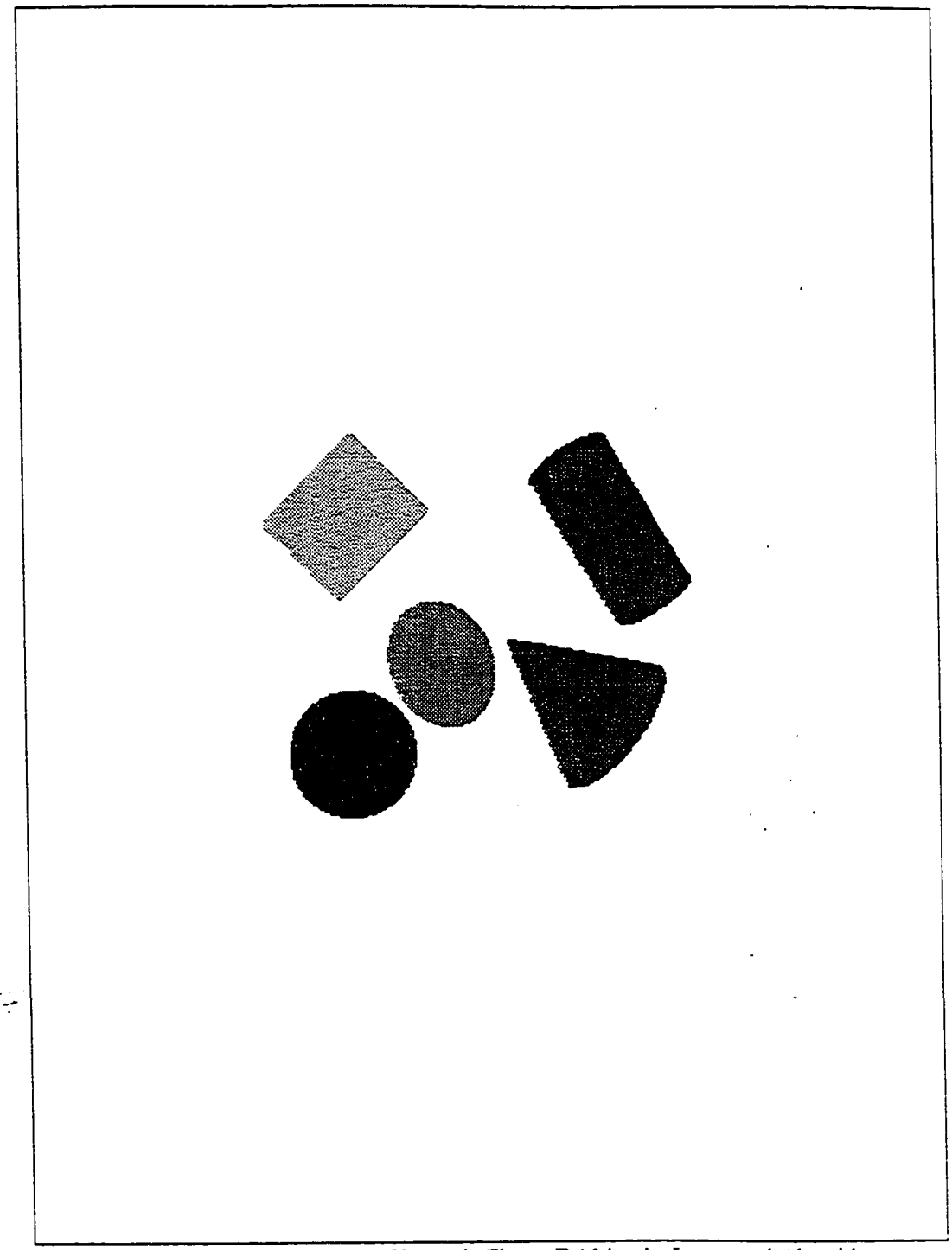
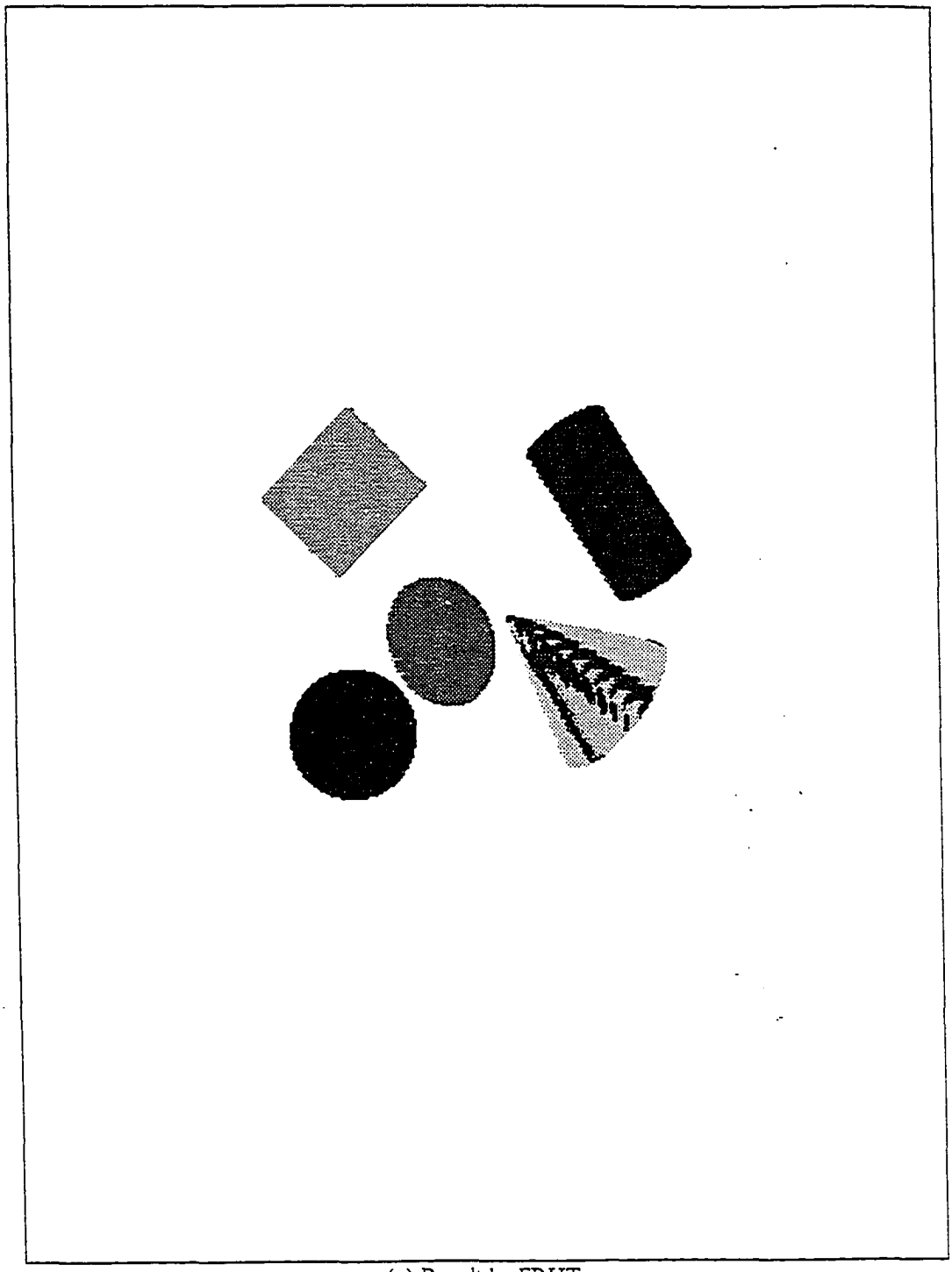
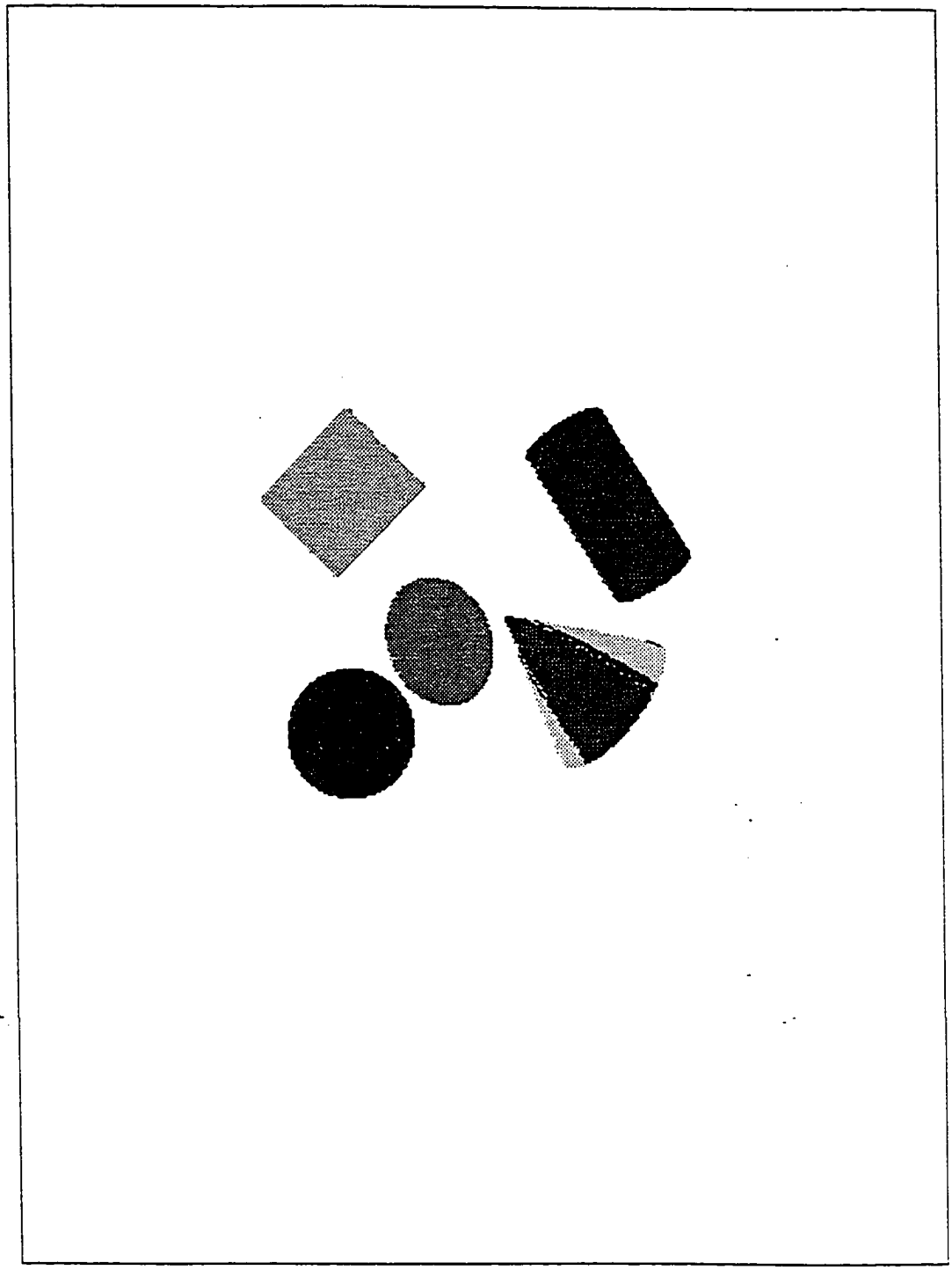


Figure E.10 Results of the Image Shown in Figure E.10 by the Integrated Algorithm.



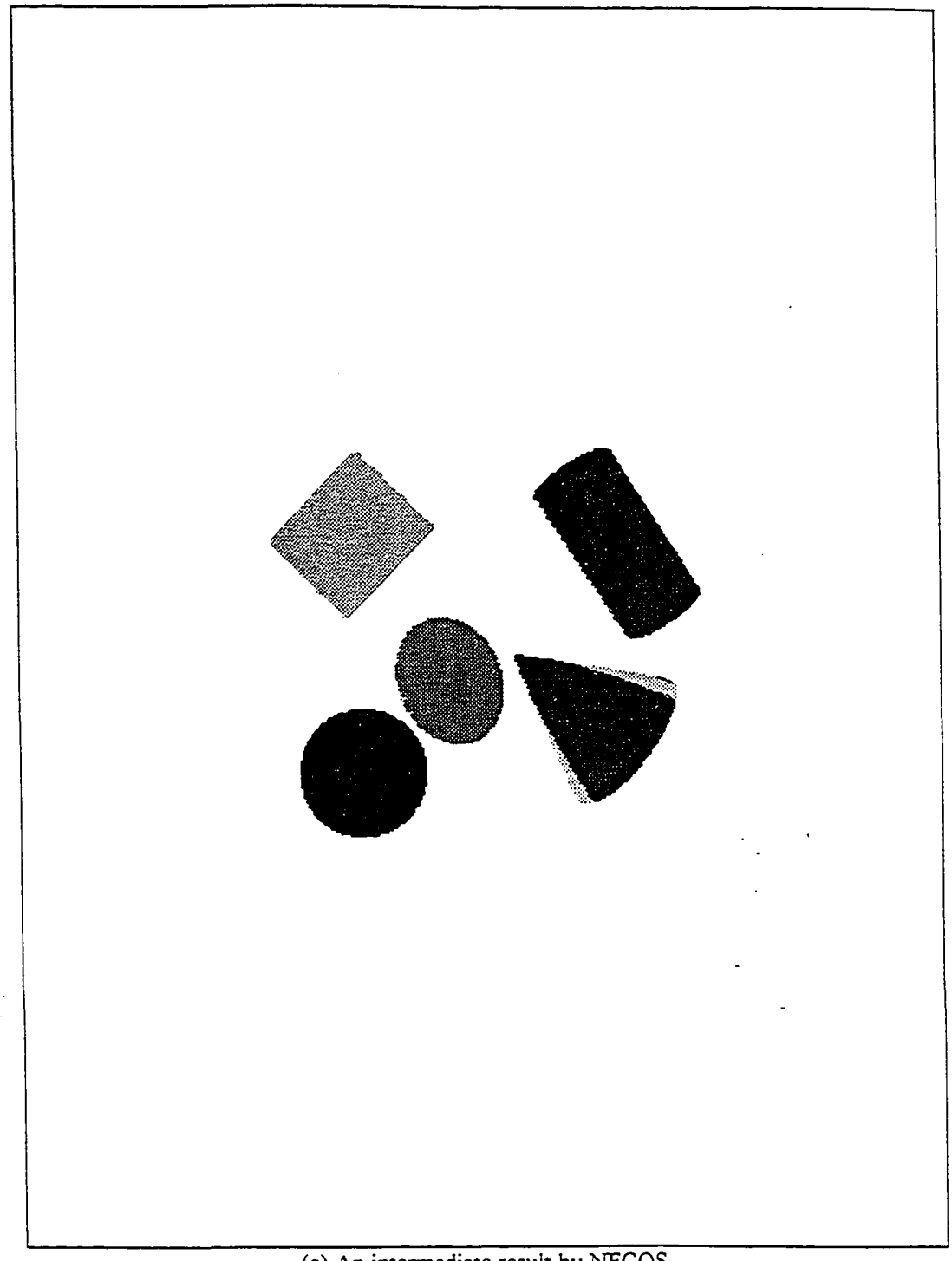
(a) Result by FRHT.

Figure E.11 The Multiple NFCQS Processes for Refining the Detection Result of the Cone Cluster.



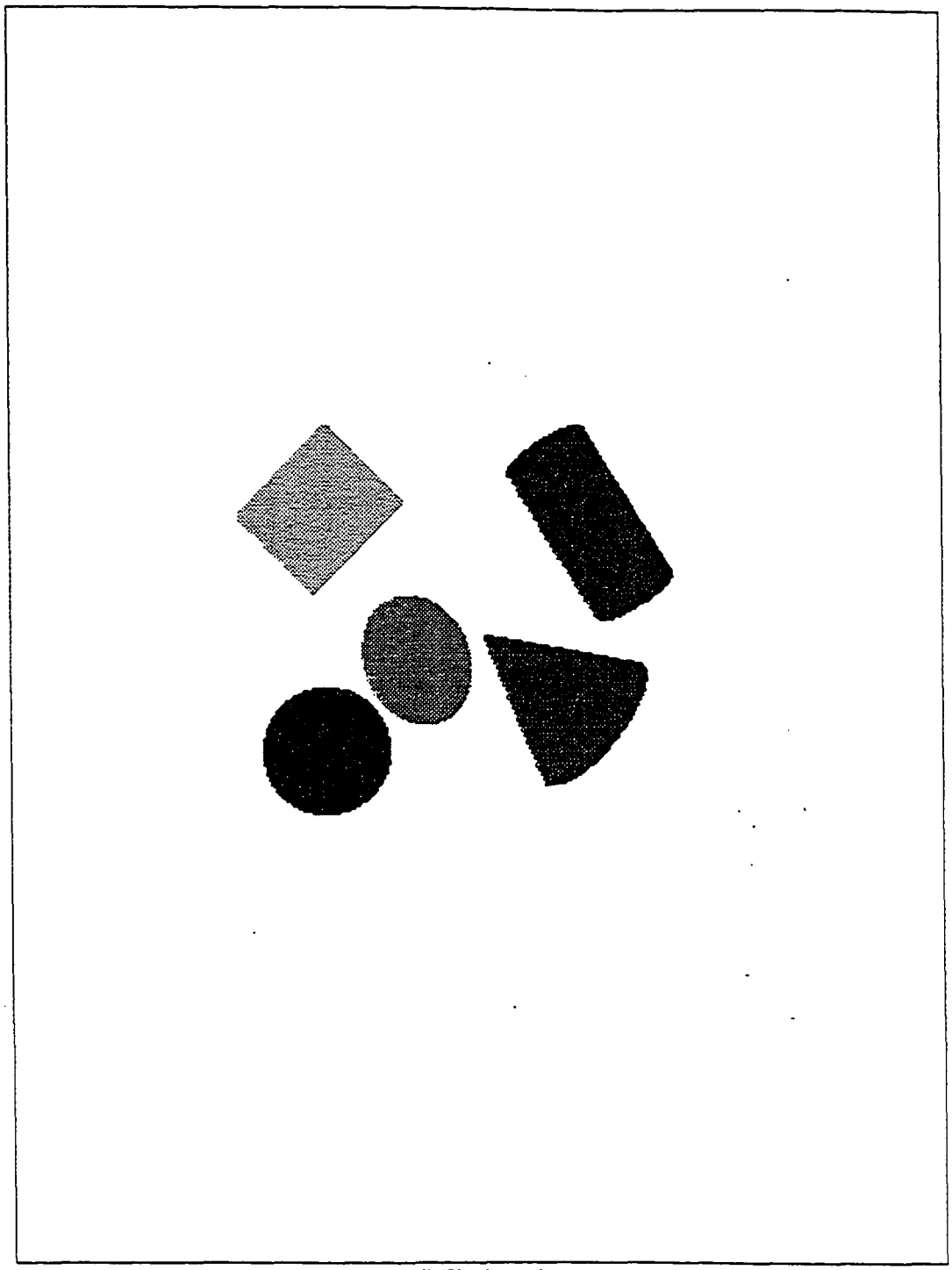
(b) An intermediate result by NFCQS.

Figure E.11 The Multiple NFCQS Processes for Refining the Detection Result of the Cone Cluster (continued).



(c) An intermediate result by NFCQS.

Figure E.11 The Multiple NFCQS Processes for Refining the Detection Result of the Cone Cluster (continued).



(d) Final result.

Figure E.11 The Multiple NFCQS Processes for Refining the Detection Result of the Cone Cluster (continued).

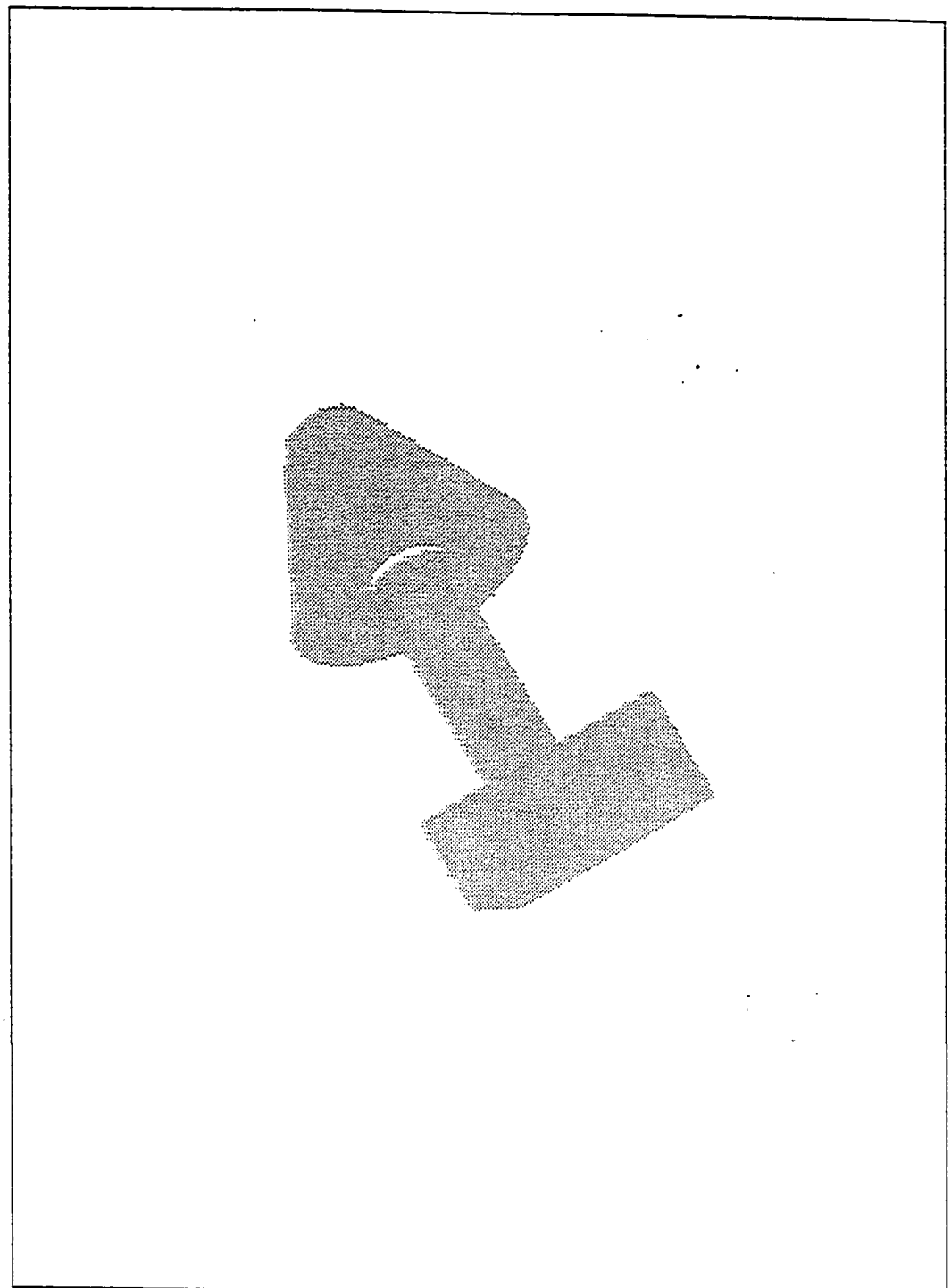


Figure E.12 A Computer Generated Range Image for a Lamp. There are 13065 points in the image and the depth values are recorded in integer numbers.

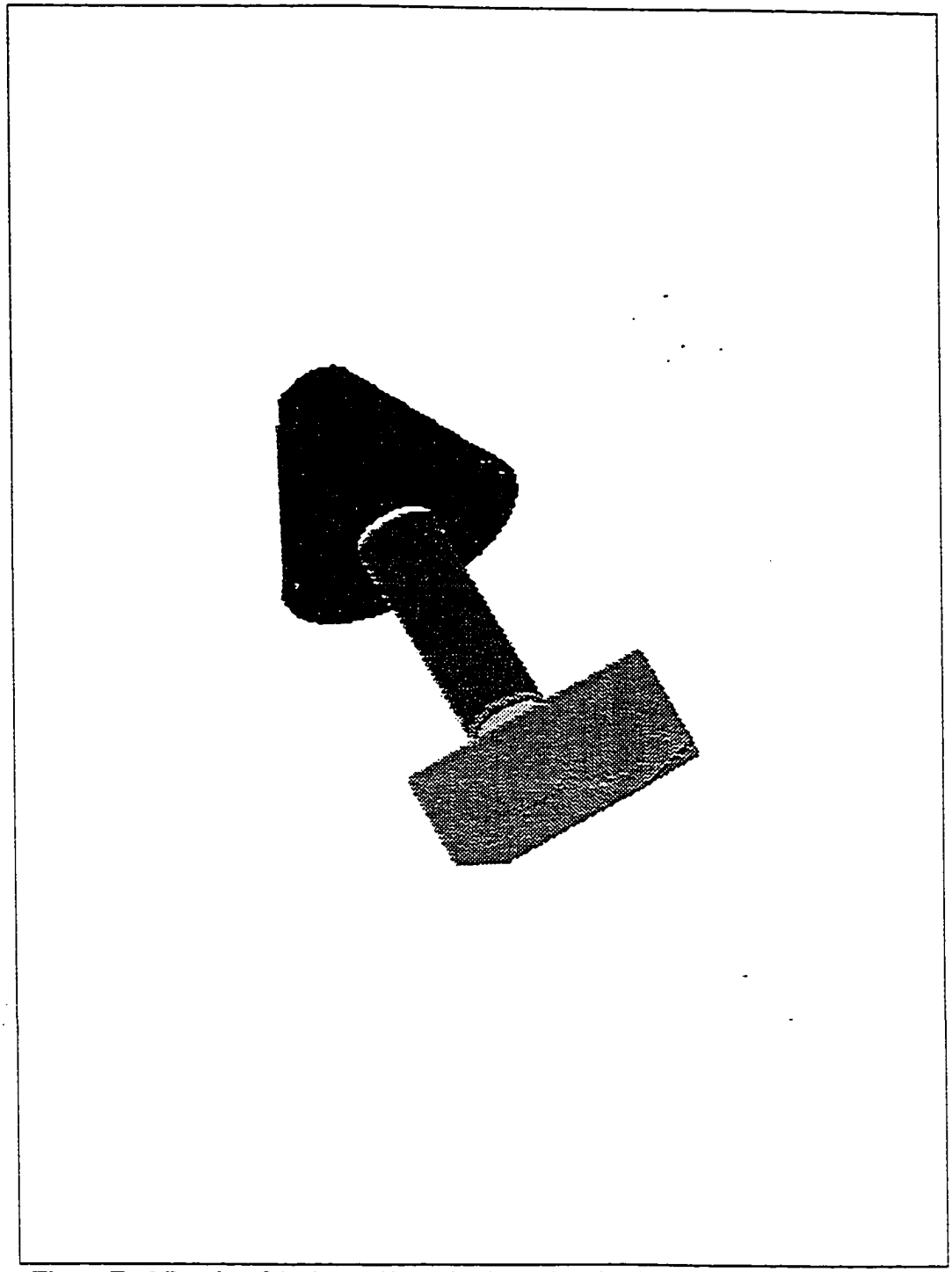


Figure E.13 Results of the Image Shown in Figure E.12 by the Integrated Algorithm.

APPENDIX F

FIGURES FOR CHAPTER 7

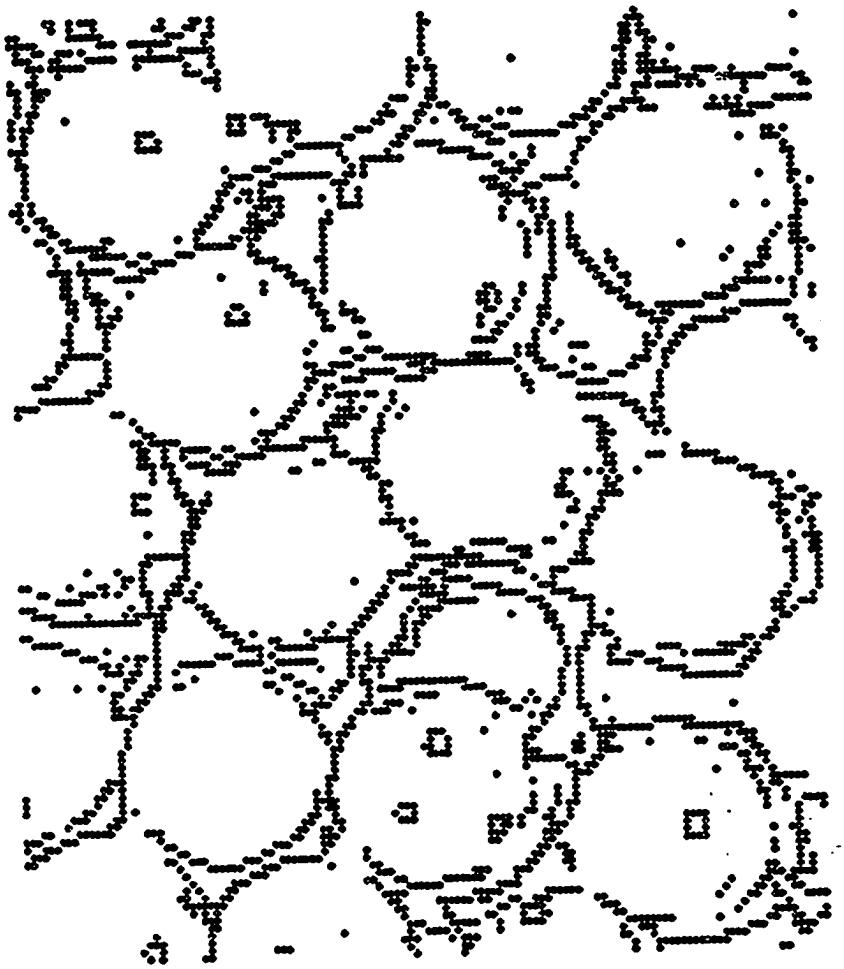


Figure F.1 Edge Data for the Image of Figure 1.1.

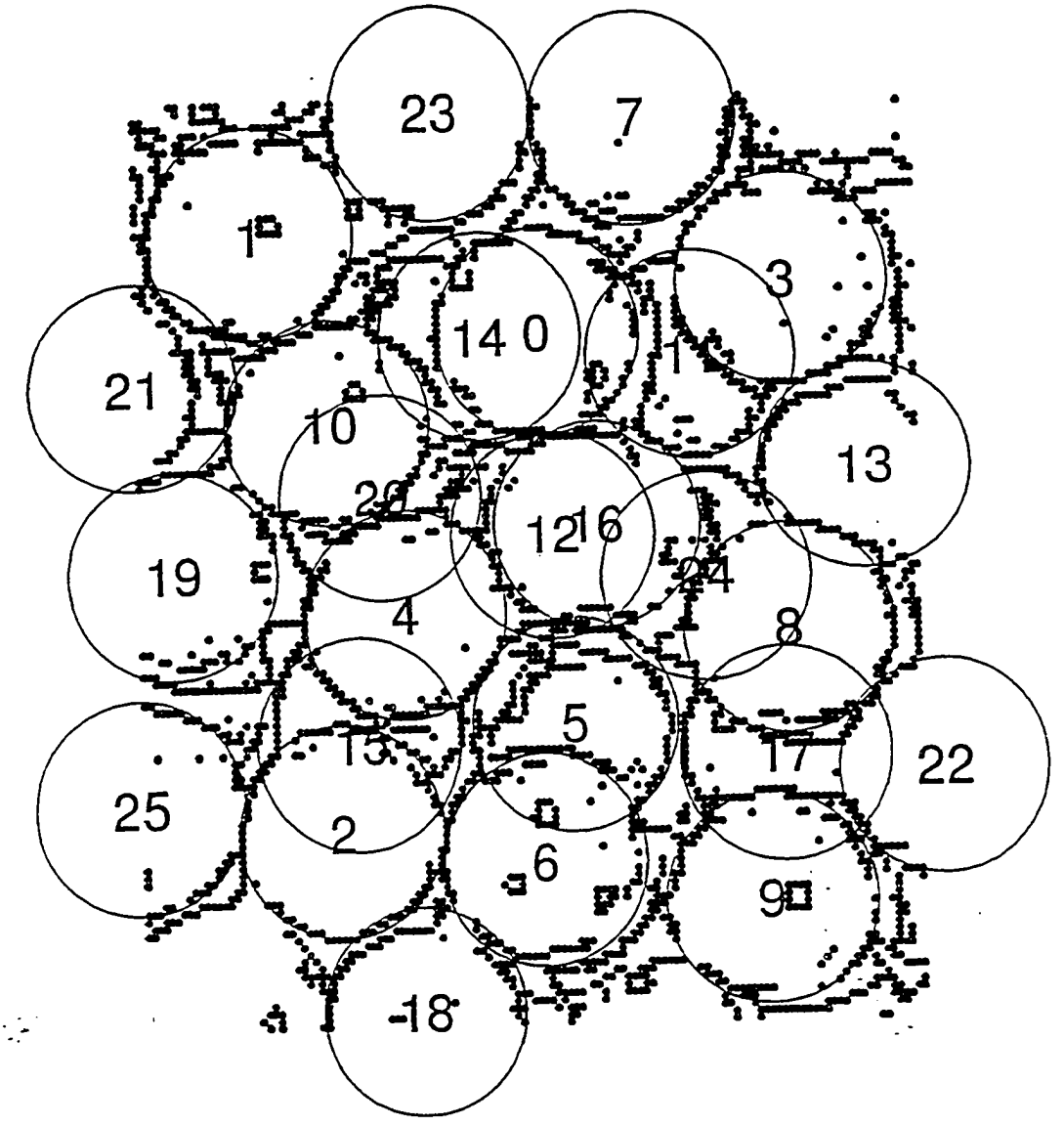


Figure F.2 Results of the Image Shown in Figure F.1 by D&C-NFCS Plotted over Edge Data for $c_{max} = 26$ (before cluster removal and merging).

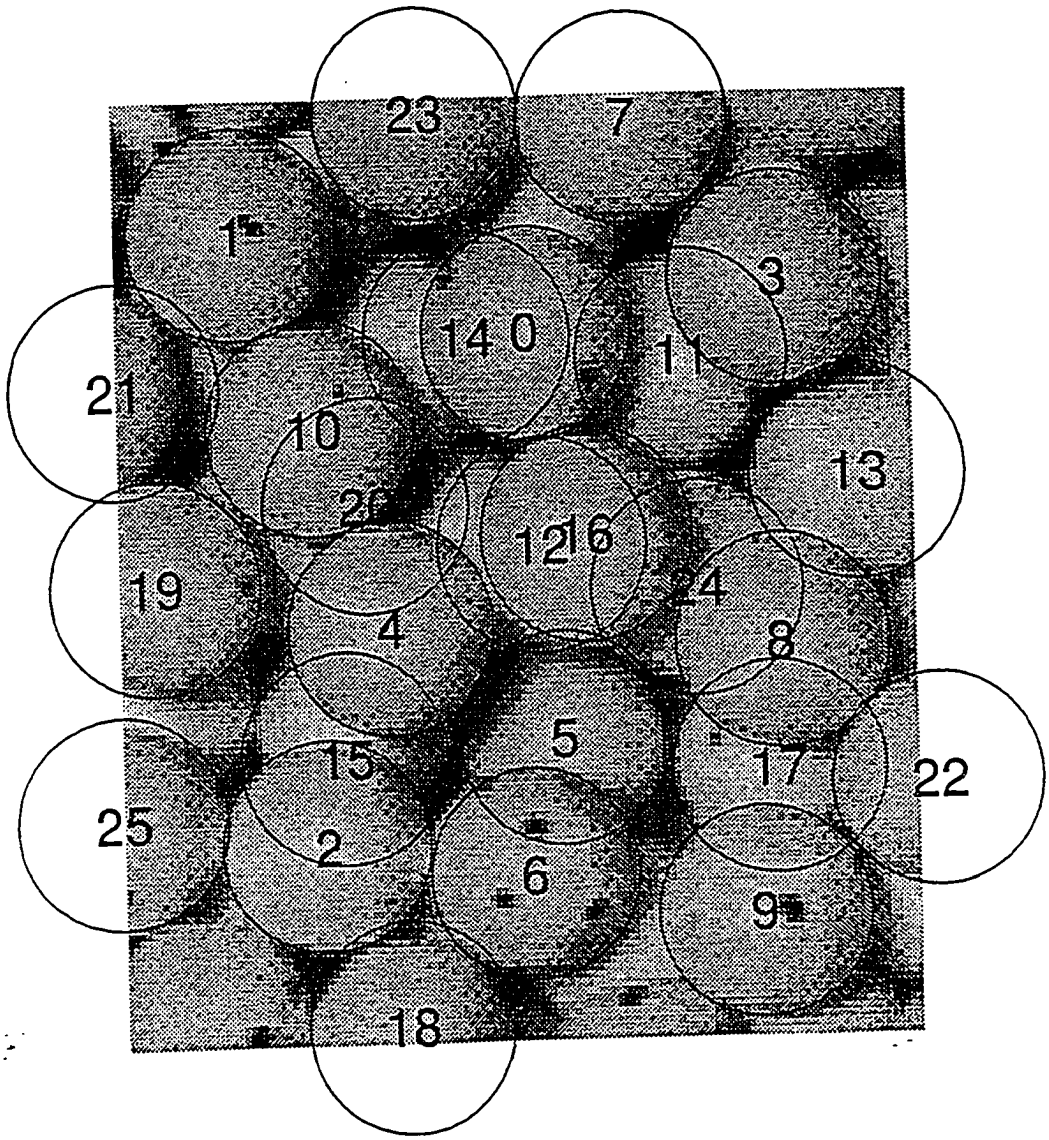


Figure F.3 Results of the Image Shown in Figure F.1 by D&C-NFCS Plotted over Original Image for $c_{max} = 26$ (before cluster removal and merging).

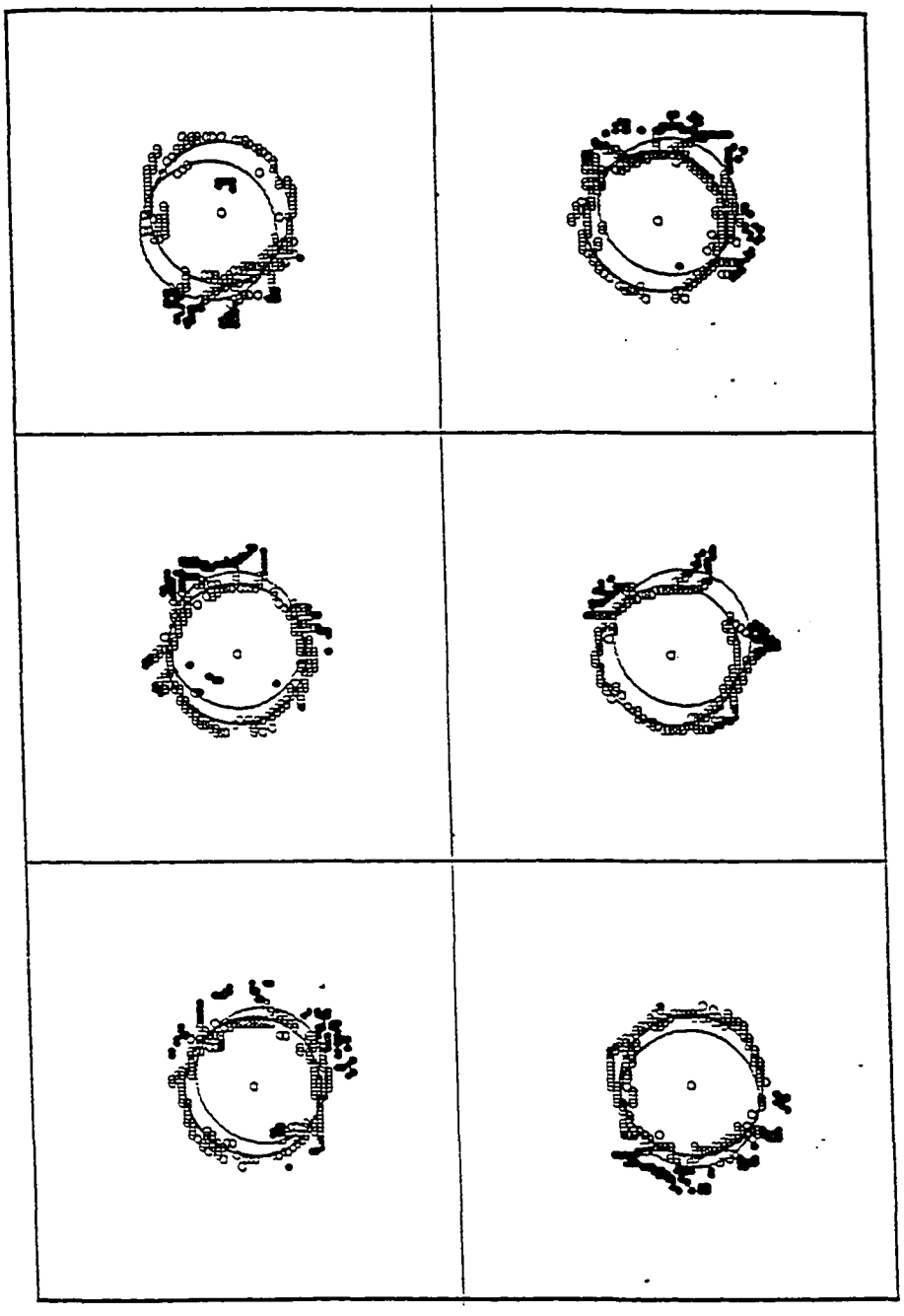


Figure F.4 Several Examples Comparing Results for Individual Cluster Detection Using NFCS and HT. Solid line circles are the results of NFCS, dashed line circles are the results of HT, small open circles are the points classified into circular cluster and the solid dots are the points classified into noise cluster.

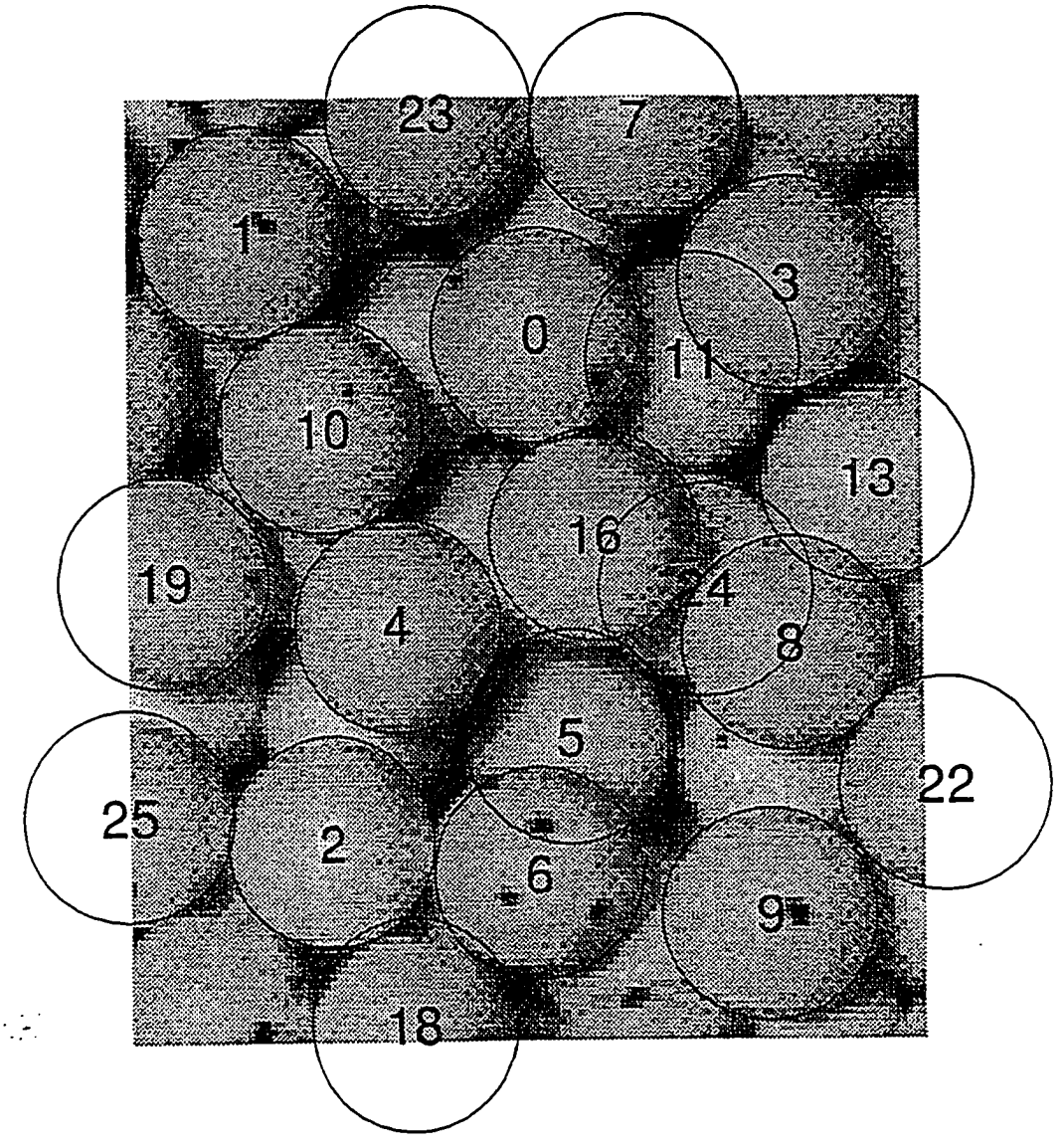


Figure F.5 Final Results of D&C-NFCS Plotted over Original Image for $c_{max} = 26$ by Using the Validity Criteria Discussed in Section 6.2.

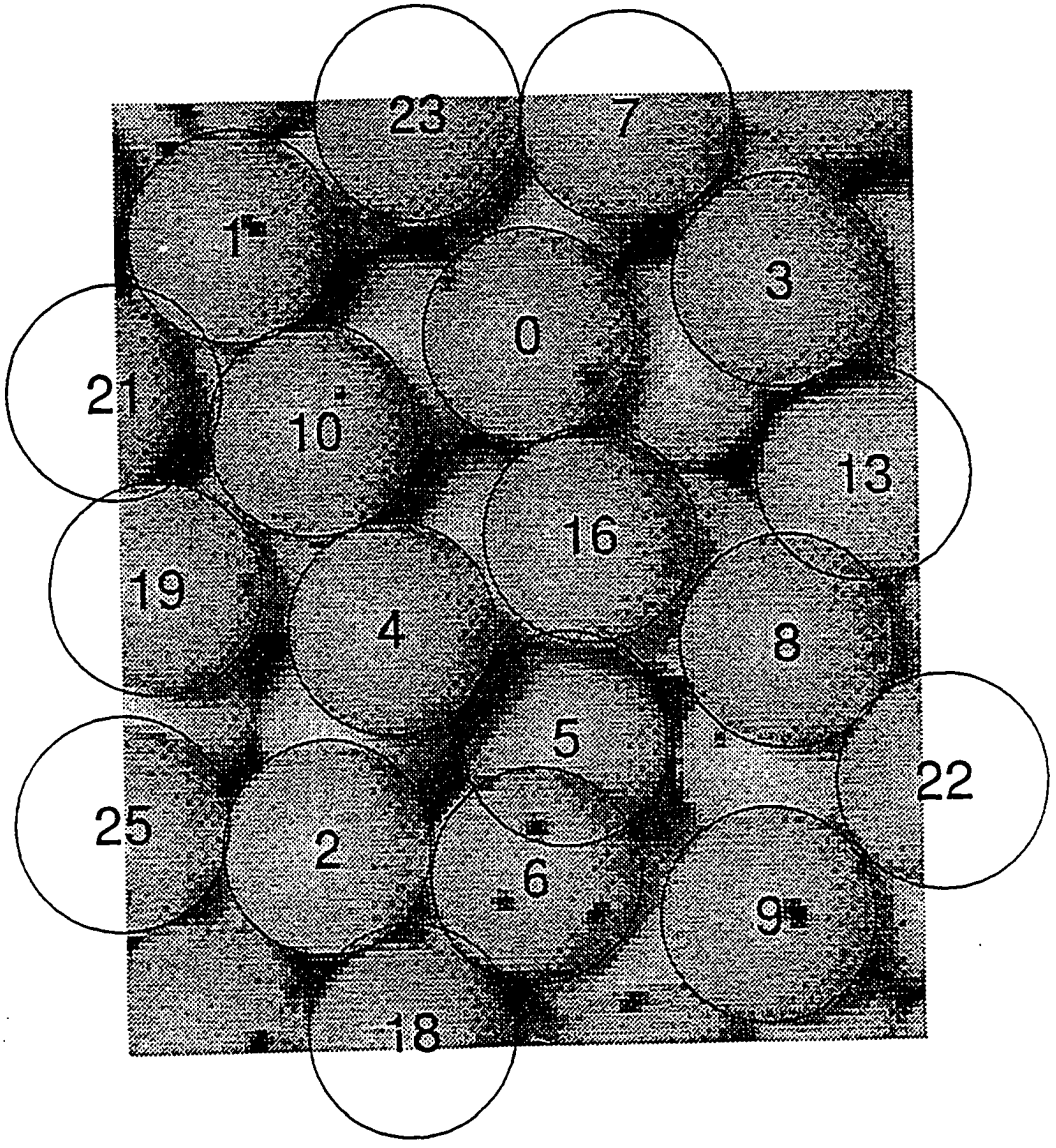


Figure F.6 Final Results of D&C-NFCS Plotted over Original Image for $c_{max} = 26$ by Using the Valid Density Criterion.

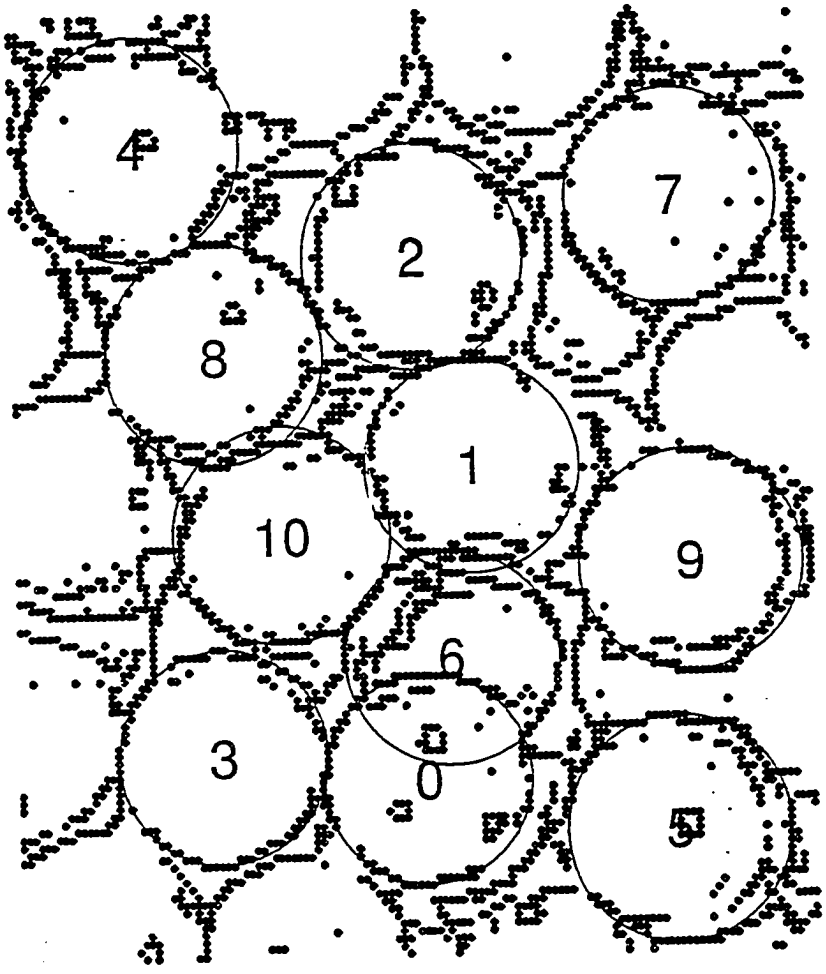


Figure F.7 Results of RHT-NFCS Plotted over Edge Data.

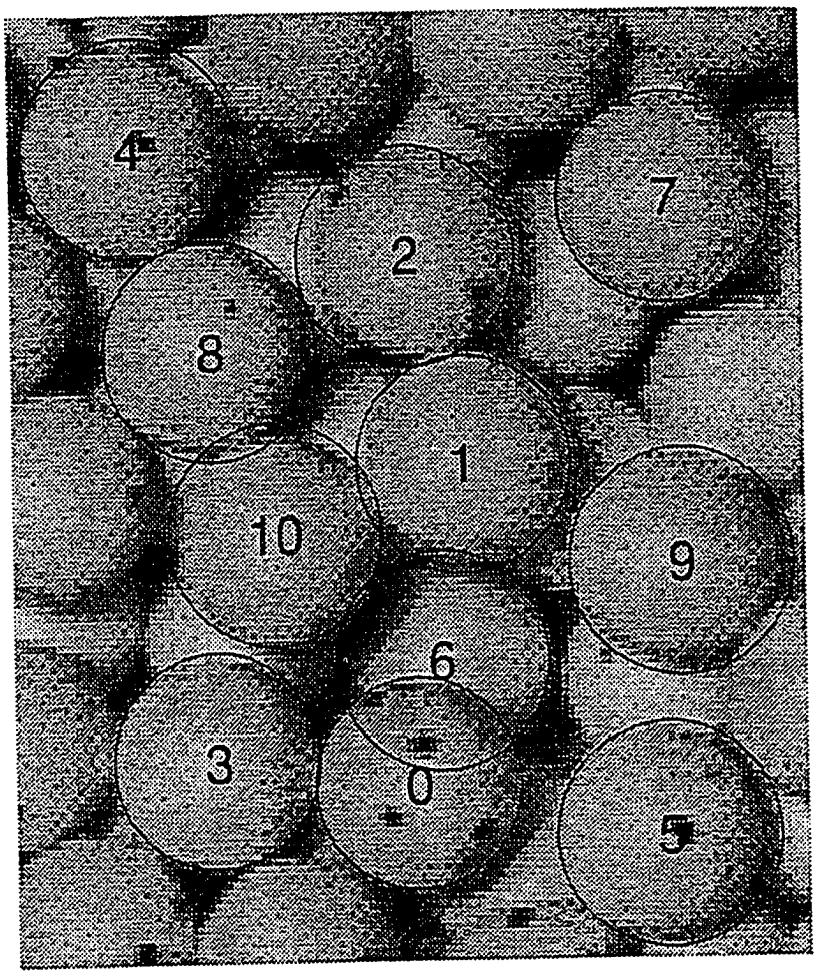


Figure F.8 Results of RHT-NFCS Plotted over Orignial Image.

REFERENCES

[1] E. Backer and A. K. Jain, "A clustering performance measure based on fuzzy set decomposition," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 3, No. 1, pp. 66-74, 1981.

[2] D. H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," *Pattern Recognition*, 13, pp. 111-122, 1981.

[3] R. Bergen and H. Shvaytser, "A probabilistic algorithm for computing Hough transforms," *Journal of Algorithms*, 12, pp. 639-656, 1991.

[4] J. Besl and R. C. Jain, "Invariant surface characteristics for 3D object recognition in range images," *Computer Vision, Graphics and Image Processing*, 33, pp. 33-80, 1986.

[5] J. C. Bezdek, "Fuzzy Mathematics in Pattern Classification," Ph.D. Thesis, Applied Math. Center, Cornell University, Ithaca, 1973.

[6] J. C. Bezdek, "Cluster validity with fuzzy sets," *J. Cybernetics*, vol. 3, pp. 58-73, 1974.

[7] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981.

[8] J. C. Bezdek, C. Coray, R. Gunderson and J. Watson, "Detection and characterization of cluster sub structure," *SIAM J. Appl. Math.*, 40, pp. 339-372, 1981.

[9] V. L. Brailovsky, "A probabilistic approach to clustering," *Pattern Recognition Letters* 12 , pp. 193-198, 1991.

[10]A. Califano, R. M. Bolle and R. W. Taylor, "Generalized neighbourhoods: a new approach to complex parameter feature extraction," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 192-199, 1989.

[11]M. Cohen, and G. T. Toussaint, "On the detection of structures in noisy pictures," *Pattern Recognition*, 9, pp. 95-98, 1977.

[12]R. N. Dave, "Use of the adaptive fuzzy clustering algorithm to detect lines in digital images," *Proceedings of Intelligent Robots and Computer Vision VIII*, Vol. 1192, 2, pp. 600-611, 1989.

[13]R. N. Dave, "Fuzzy shell-clustering and applications to circle detection in digital images," *International J. of General Systems*, 16(4), pp. 343-355, 1990.

[14]R. N. Dave, "Characterization and detection of noise in clustering," *Pattern Recognition Letters*, Vol. 12(11), pp. 657-664, 1991a.

[15]R. N. Dave, "New measures for evaluating fuzzy partitions induced through c-shells clustering," *Proc. of the SPIE Conference on Intelligent Robot and Computer Vision X: SPIE* vol. 1607, Boston, pp. 406-414, Nov. 1991b.

[16]R. N. Dave, and S. Bhamidipati, "Application of fuzzy shell-clustering algorithm to recognize circular shapes in digital images," *Proceedings of Third Annual IFSA World Congress*, pp. 238-241, 1989.

[17]R. N. Dave, and K. Bhaswan, "Adaptive fuzzy c-shells clustering and detection of ellipses," *IEEE Trans. on Neural Networks*, Vol. 3(5), 1992.

[18]R. N. Dave and T. Fu, "Robust shape detection using fuzzy clustering: practical applications," *Fuzzy Sets and Systems*, August 1994.

[19]R. N. Dave and K. J. Patel, "Progressive fuzzy clustering algorithms for characteristic shape recognition," *Proceedings of NAFIP90*, pp. 121-124, 1990a.

[20]R. N. Dave and K. J. Patel, "Fuzzy ellipsoidal-shell clustering algorithm and detection of elliptical shapes," *Proceedings of Intelligent Robots and Computer Vision IX: Algorithms and Techniques*, Vol. 1381, pp. 320-333, 1990b.

[21]R. N. Dave, J. Yu and A. D. Rosato, "Measurement of collisional properties of spheres using high-speed video analysis," *1993 ASME WAM: Symposium on Mechatronics*, DSC-vol. 50, pp. 217-222, 1993.

[22]D. Dimitrescu, "Hierarchical pattern classification," *Fuzzy Sets and Systems*, vol. 28, pp. 145-162, 1988.

[23]R. Duda and P. Hart, "Use of the Hough transform to detect lines and curves in pictures," *Communications of the Association of Computing Machinery*, 15, pp. 11-15, 1972.

[24]R. Duda and P. Hart, *Pattern Classification and Scene Analysis*, John Wiley, 1973.

[25]B. S. Everitt, *Cluster Analysis*, Wiley, New York, 1974.

[26]S. Foerster, M. Y. Louge, H. Chang and K. Allia, "Measurement of the collision properties of small spheres," *Physics of Fluids*, Vol. 6, (to appear 1994).

[27]T. Fu and R. N. Dave, "Segmenting curved surfaces from range images," *Ninth International Conference on CAD/CAM, Robotics, and Factories of the Future, ISPE*, August 1993.

[28]I.Gath and A. B. Geva, "Unsupervised optimal fuzzy clustering," *IEEE Trans. PAMI.* vol. 11, No. 7, pp. 773-781, 1989.

[29]C. Ghidaglia, E. Guazzelli, L. de Arcangelis, and L. Oger, "Particulate transport in consolidated granular systems," in: C. Thornton (ed.), *Powder & Grain* 93, Balkema, Rotterdam, pp. 389-393, 1993.

[30]M. P. Groover and E. W. Zimmers, Jr., *CAD/CAM: Computer-Aided Design and Manufacturing*, Prentice-Hall, New Jersey, 1984.

[31]E. E. Gustafson and W. C. Kessel, "Fuzzy clustering with a fuzzy covariance matrix," *Proceedings of IEEE CDC*, San Diego, Calif., pp. 761-766, 1979.

[32]D. Hanes, "Observations of granular flow in an inclined chute," *Fifth NSF-DOE Workshop on Flow of Particulates and Fluids*, pp. 257-264, 1993.

[33]R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*, Addison-Wesley, 1992.

[34]R. Hoffman and A. K. Jain, "Segmentation and classification of range images," *IEEE Trans. PAMI.*, vol. PAMI-9, No. 5, pp. 608-620, 1987.

[35]P. V. C. Hough, "A method and means for recognizing complex patterns," U.S. Patent No. 3069654, 1962.

[36]S. S. Hsiau and M. L. Hunt, "Shear-induced particle diffusion and longitudinal velocity fluctuations in a granular flow mixing layer," to appear in *J. Fluid Mechanics*, 1993.

[37]J. Illingworth and J. Kittler, "The adaptive Hough transform," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 9, pp. 690-698, 1987.

[38]J. Illingworth and J. Kittler, "A survey of Hough transforms," *Computer Vision, Graphics and Image Processing*, 44, pp. 87-116, 1988.

[39]A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, *Prentice-Hall*, Englewood Cliffs, New Jersey, 1988.

[40]J. Jolion and A. Rosenfeld, "Cluster detection in background noise," *Pattern Recognition*, 22(5), pp. 603-607, 1989.

[41]J.-M. Jolion, P. Meer, and S. Bataouche, "Robust clustering with applications in computer vision," *IEEE Trans. on PAMI*, Vol. 13, No. 8, pp. 791-802, 1991.

[42]D. Keren, D. Cooper, and J. Subrahmonia, "Describing complicated objects by implicit polynomials," *IEEE Trans. on PAMI*, Vol. 16, NO. 1, pp. 38-53, 1994.

[43]C. Kimme, D. H. Ballard and J. Sklansky, "Finding circles by an array of accumulators," *Communications of the Association of Computing Machinery*, 18, pp. 120-122, 1975.

[44]N. Kiryati, Y. Eldar and A. Bruckstein, "A probabilistic Hough transform," *Pattern Recognit.* 24(4), pp. 303-316, 1991.

[45]G. Krishnan, I. Rampall and D. T. Leighton, Jr., "Particle dynamics near a solid wall in concentrated suspensions of non-colloidal spheres," *Fourth NSF-DOE Workshop on Flow of Particulates and Fluids*, pp. 79-92, 1992.

[46]R. Krishnapuram, H. Frigui and O. Nasraoui, "A fuzzy clustering algorithm to detect planar and quadric shpaes," *Proceedings of the North American Fuzzy Information Processing Society Workshop*, Puerto Vallarta, Mexico, December 1992.

[47]R. Krishnapuram, H. Frigui and O. Nasraoui, "Fuzzy and possibilistic shell clustering algorithms and their application to boundary detection and surface approximation: part I," Department of Electrical and Computer Engineering, University of Missouri-Columbia, 1993a.

[48]R. Krishnapuram, H. Frigui and O. Nasraoui, "Fuzzy and possibilistic shell clustering algorithms and their application to boundary detection and surface approximation: part II," Department of Electrical and Computer Engineering, University of Missouri-Columbia, 1993b.

[49]R. Krishnapuram, H. Frigui and O. Nasraoui, "Quadratic shell clustering algorithm and the detection of second-degree curves," *Pattern Recognition Letters*, vol. 14, No. 7, pp. 545-552, July 1993c.

[50]R. Krishnapuram, O. Nasraoui and H. Frigui, "The fuzzy c spherical shells algorithms: a new approach," *IEEE Trans. on Neural Networks*, vol. 3, No. 5, pp. 663-671, Sept. 1992.

[51]V. F. Leavers, "The dynamic generalized Hough transform: its relationship to the probabilistic Hough transforms and an application to the concurrent detection of circles and ellipses," *CVGIP: Image Understanding*, Vol. 56, No. 3, November, pp. 381-398, 1992.

[52]A. Leroy and P. J. Rousseeuw, "PROGRESS: a program for robust regression Analysis," *Technical Report 201, Center for Statistics and O.R.*, University of Brussels, Belgium, 1984.

[53]H. Li, M. A. Lavin and R. J. LeMaster, "Fast Hough transform: a hierarchical approach," *CVGIP*, 36, pp. 139-161, 1986.

[54]X. Li, W. C. Dass and C. W. Manzione, "Characterization of internal microstructure of granular materials using computerized tomography," *Proc. of ASME WAM-92*, MD-vol. 37, pp. 107-113, 1992.

[55]P. M. Merlin and D. J. Farber, "A parallel mechanism for detecting curves in pictures," *IEEE Trans. on Computers*, 24, pp. 96-98, 1975.

[56]A. D. Rosato, Private communication, 1993.

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

[58]P. J. Rousseeuw, "Multivariate estimation with high breakdown point," *Fourth Pannonian Symposium on Mathematical Statistics and Probability*, Bad Tazmannsdorf, Austria, Sept. 4-9, 1983.

[59]P. J. Rousseeuw, "Least median of squares regression," *J. Am. Stat. Assoc.*, 79, pp. 871-880, 1984.

[60]P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection*, John Wiley & Sons, 1987.

[61]H. C. Schildt,*The Complete Reference*, McGraw-Hill, 1988.

[62]S. D. Shapiro, "Transformations for the computer detection of curves in the noisy pictures," *Computer Graphics and Image Processing*, 4, pp. 328-338, 1975.

[63]S. D. Shapiro, "Feature space transforms for curve detection," *Pattern Recognition*, 10, pp. 129-143, 1978a.

[64]S. D. Shapiro, "Transform method of Curve detection for textured image data," *IEEE Trans. on Computers*, 27, pp. 254-255, 1978b.

[65]S. D. Shapiro, "Properties of transforms for the detection of curves in noisy images," *Computer Graphics and Image Processing*, 8, pp. 219-236, 1978c.

[66]S. D. Shapiro and A. Iannino, "Geometric constructions for predicting Hough transform performance," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1, pp. 310-317, 1979.

[67]T. M. Silberberg, L. Davis and D. Harwood, "An iterative Hough procedure for three-dimensional object recognition," *Pattern Recognition*, Vol. 17, No. 6, pp.621-629, 1984.

[68]T. M. Silberberg, D. A. Harwood, and L. S. Davis, "Object recognition using oriented model points," *CVGIP*, 35, pp.47-71, 1986.

[69]G. Taubin, "Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with application to edge and range sgmentation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 13, No. 11, pp. 1115-1138, 1991.

[70]R. L. Thorndike, "Who belongs in the family," *Psychometrika*, vol. 18, pp. 267-276, 1953.

[71]R. Y. Tsai and T. S. Huang, "Uniqueness and estimation of three dimensional motion parameters of rigid objects with curved surfaces," *IEEE Trans. Pattern Anal. and Machine Intellignece*, Vol, PAMI-6, pp. 13-27, 1984.

[72]H.-T. Tsui and C.-K. Chan, "Hough techniques for 3D object recognition," *IEE Proceedings, Part E, Computers and Digital Techniques*, V136, pp. 565-568, 1989.

[73]S. Tsuji and F. Matsumoto, "Detection of ellipses by modified Hough transformation," *IEEE Trans. on Computers*, 27, pp. 777-781, 1978.

[74]H. Tsukune and K. Goto, "Extracting elliptical figures from an edge vector field," *IEEE Computer Vision and Pattern Recognition Conf.*, Washington, pp. 138-141, 1983.

[75]T. M. Van Veen. and F. C. A. Groen, "Discretization errors in the Hough transform," *Pattern Recognition*, 14, pp. 137-145, 1981.

[76]H. Wechsler and J. Sklansky, "Automatic detection of ribs in chest radiographs," *Pattern Recognition*, 9, pp. 21-30, 1977.

[77]I. Weiss, "Straight line fitting in a noisy image," *Proc. Computer Vision and Pattern Recognition*, pp. 647-652, 1988.

[78]M. P. Windham, "Cluster validity for fuzzy clustering algorithms," *Fuzzy Sets and Systems*, vol. 5, pp. 177-185, 1981.

[79]L. Xu, E. Oja and P. Kultanen, "A new curve detection method: randomized Hough transform (RHT)," *Pattern Recognition Letters*, 11, pp. 331-338, May 1990.

[80]J. Yu, "Evaluation of Collision Properties of Spheres Using High-Speed Video Analysis," M.S. Thesis, New Jersey Institute of Technology, 1993.