ON EXTENDING THE GENERALIZED HOUGH TRANSFORM

by

Benguang Yao

B.Sc., University of Science and Technology of China, 1985

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

in the School
of
Computing Science

SIMON FRASER UNIVERSITY
July 1990

Approval

Name: Benguang Yao
Degree: Master of Science
Title of Thesis: On Extending the Generalized Hough Transform

William S. Havens
Chairman

Ze-Nian Li
Senior Supervisor

Thomas W.
Supervisor

Kamal Gupta
External Examiner

Aug. 5, 1990
Date Approved

PARTIAL COPYRIGHT LICENSE

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

Title of Thesis/Project/Extended Essay

On Extending the Generalized Hough Transform.

Author: _____
(signature)

Benguang Yao
(name)

August 8, 1990
(date)

ABSTRACT

Matching models to images is a very important task in image understanding. The generalized Hough transform (GHT) proposed by Ballard has been proven to be a very effective method for matching models of arbitrary shapes to images. It converts the problem of global pattern detection into a problem of local peak finding. However, the GHT has difficulty handling images with occlusions due to the reduced peak values when objects are partially occluded. It can easily make mistakes when images contain patterns similar to the model. Moreover, it is also very difficult to apply parallel processing. The unique peak spot results in contention when processors try to access it. In this thesis, An improved version of Ballard's GHT is proposed which provides a potentially more robust and systematic technique, the linear Hough transform, for solving the problems in the detection of partially occluded objects. We use a linear numeric pattern to replace the peak in the GHT and use the relationship between entries in the linear pattern to achieve high robustness. Partial matches of the linear pattern are used for partial object detection. Finally, we present a parallel version of our new technique to exploit the parallel computational power of array processors. High performance is achieved by taking advantage of the speed-up due to reduced contention in the accumulation process which results with our linear numeric pattern.

ACKNOWLEDGEMENTS

I would like to thank my senior supervisor, Dr. Ze-Nian Li, for supervising this thesis. He led me into the area of *computer vision*. His teaching on *computational vision*, his support, guidance and encouragement of my work, and especially his extraordinary patience while modifying and editing my thesis drafts have made this thesis a reality. I would like to thank the other member of my thesis supervisory committee, Dr. Thomas Calvert, for the valuable time he spent answering my questions. His extraordinary patience in modifying and editing my original version of the thesis and his constructive comments have helped improve greatly the original version of the thesis. I am indebted to Dr. Kamal Gupta for being my external examiner and for his constructive comments on the original version of the thesis.

I would like to thank all the professors in the School of Computing Science who have led me through various ares of computer science in the past few years.

I would like to thank all of my friends at S.F.U. Their encouragement and help enabled me to overcome difficulties, and made my stay at S.F.U. very memorable. In particular, Hong Fan has given me much encouragement, support, and timely help when I met with difficulties. Frank Tong helped me on my conference paper and this thesis. He has always been a source of help when I needed it. I would also like to acknowledge Janis Harper for her professionally editing the written text of this thesis and John Norman for reading early version of this thesis.

I would like to thank Simon Fraser University for its financial support of my study through fellowship and teaching assistantship.

Finally, I would like to thank my parents for their love, education and support.

CONTENTS

CHAPTER 1

INTRODUCTION

Vision is a powerful sense for human beings. Through it, we are able to identify objects and learn their positions and thus collect tremendous amounts of information about our surrounding environment without any direct physical contact. It is not surprising that people have been trying to equip machines with the visual sense by using modern computing technology.

Although we still know very little about how images are processed in the human vision system, some very significant progress has been made in developing computer vision systems. Many modern machines are equipped with different level vision systems which have been operated quite successfully in various environments. For example, an inspection robot arm that can selectively pick or separate parts on a conveyor belt in an assembly line (as shown in Figure 1.1) is a typical industrial application of computer vision technology.

In industrial automation, there is a great interest in recognizing parts even if they are partially occluded. Although it is possible to use shakers and some custom machinery to separate, pelletize, or prearrange the parts for easy recognition, it is more flexible and convenient to use a vision system that can recognize the parts even though they may be partially occluded or placed in an arbitrary orientation.

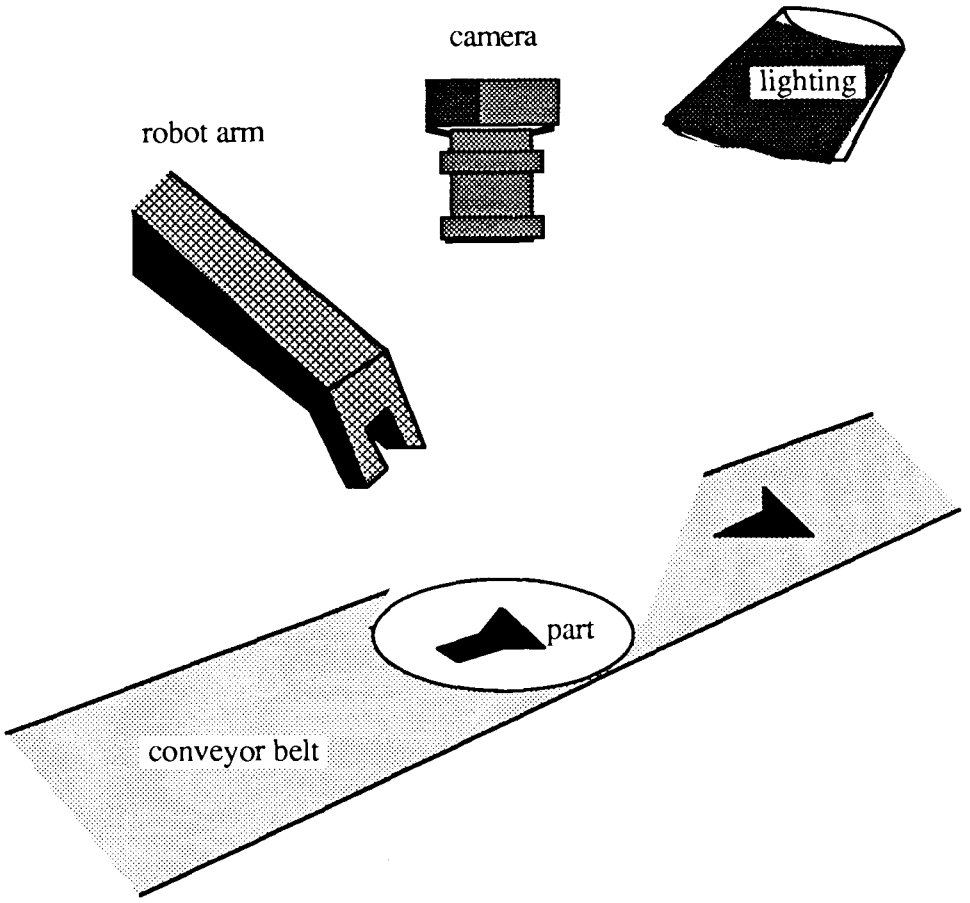


Figure 1.1 Robot Arm Picking Parts On Assembly Line

A very important task in image understanding is *scene analysis*. Its goal is to construct symbolic scene descriptions by extracting information from images or image sequences. Detecting the occurrences of objects with known shapes is a very essential part of the information extraction process.

Figure 1.2 describes an industrial vision system and the links within the control unit of an industrial robot arm. The goal of such a vision system is to give a symbolic description of what is in the scene of the image. The machine, in this case a robot arm

equipped with the vision system, can be directed and can act intelligently in its environment by using the symbolic description.

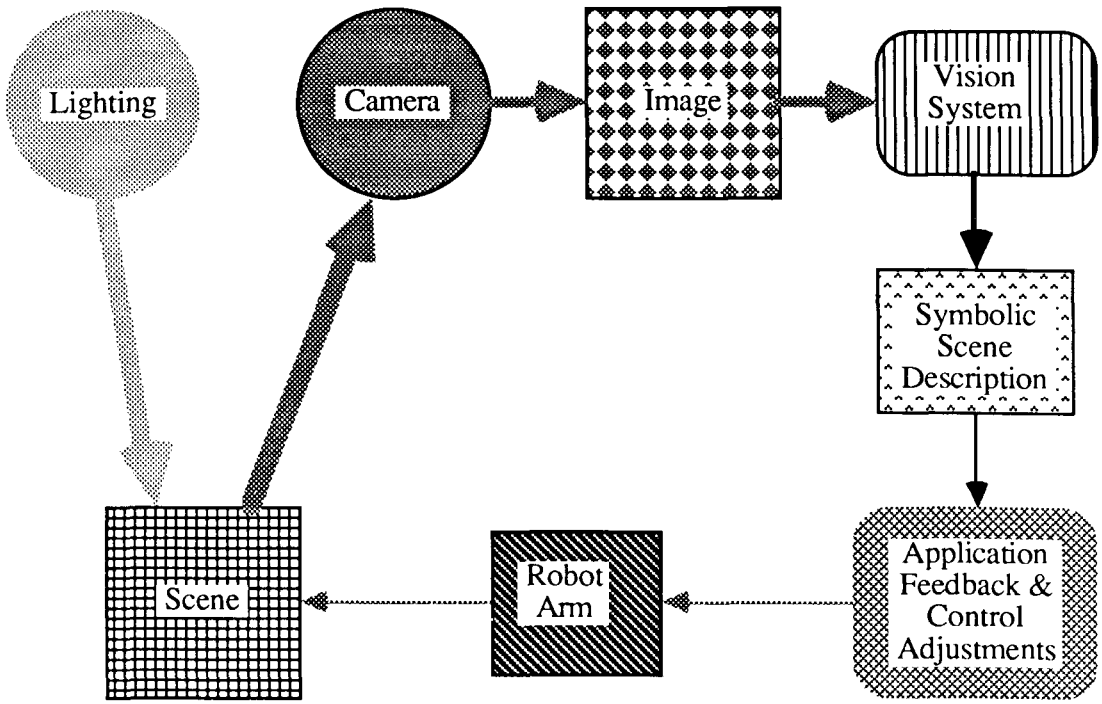


Figure 1.2 Links In An Industrial Robot Arm and Its Control Unit

In a typical industrial application, the objects to be detected may not have well-defined analytical boundaries. Frequently, the model objects are described by templates. The obvious approach to solve this object detection problem is to use template matching [Duda73]. Some other early attempts to match a model template with the edge points in images have been suggested (e.g., Perk78, Perk80). But the slow speed of performing template matching has led to a number of investigations aimed at increasing its efficiency.

Several early methods have been studied (e.g., Nage72, Vand77). Ballard showed how to use the Generalized Hough Transform (GHT) to solve the problem [Ball81].

The Hough Transform (HT) was initially proposed for detecting straight lines [Houg62]. The general idea of the Hough transform is to transfer the problem of a global pattern detection in image space into the easier to solve local peak detection problem in parameter space. This is achieved by determining specific values of parameters which characterize the line. Rosenfeld [Rose69], and Duda and Hart [Duda72] improved the technique and used it to detect circles. Kimme et al [Kimm75] and Tsuji et al [Tsuj78] extended it to detect ellipses. Ballard [Ball81] generalized the technique and showed how to use it to detect objects with arbitrary shapes.

Ballard's Generalized Hough Transform (GHT) acquired considerable computational efficiency by taking full advantage of gradient orientation information of edge points. However, the GHT does not readily suggest the solution when the object sought was incomplete or partially occluded.

There have been some studies on extending Ballard's work to detect partially-occluded objects. A typical approach is to apply the GHT on sub-object models compounded with multiresolution techniques [Davi82]. However, this type of approach lacks a systematic decomposition standard in dividing the object model into its sub-object models. Therefore, it is not a well defined approach for solving the problem.

On the other hand, the GHT itself inherits some robustness problems in the sense of its effectiveness of finding clusters of similar transforms in order to match a model to an

image [Grim88], since noise and computational errors may transfer a similar pattern in an image into a peak in the parameter space which may be erroneously picked up by the GHT.

Parallel computation has been attracting more and more interest among computing scientists as an approach to improve the speed of many kinds of computations. In computer vision, it is most desirable to build a super-fast vision system to meet the time requirement for many real applications. A vision system mounted onto a mobile robot is a typical example of an application of a real-time vision system (it is not desirable for a mobile robot to stand still for 20 to 30 minutes for just one image frame analysis to finish).

The increasingly important role played by the Hough transform in image understanding makes improvement in its speed very meaningful. There have been many reports on parallel computation of Hough transforms for straight lines (e.g., Li90, Cyph87, Rose88, Ibra86). But due to the irregularities of the object models dealt by GHT and the irregularities in performing the generalized Hough transform, the GHT can not readily handle the potential transform contentions in parameter space which result from the parallel processing of image points.

This thesis studies a systematic approach to extending the GHT to detect partially-occluded objects and to improving the robustness of the GHT; there is also an exploration of parallel computation applied to the extended GHT. The thesis is organized into six chapters. Chapter Two contains a brief survey of the Hough transform technique and its applications. In Chapter Three, we present a new technique called the Linear Generalized Hough Transform which is more robust than the GHT and which is also capable of

detecting partially-occluded parts. Later in the chapter, we incorporate the V-GHT algorithm into a more general algorithm, the AV-GHT algorithm, to deal with more general cases. In Chapter Four, we explore the parallel computational power of a linear array processor when applied to our Linear GHT technique and the V-GHT algorithm developed in Chapter Three. Two parallel algorithms and an analysis of their efficiency are presented in this chapter. The first algorithm is based on a basic linear array processor architecture. and the second uses more advanced linear array architectures and demonstrates some techniques for resolving contentions. We also discuss the potential for a more powerful hardware architecture. In Chapter Five, we present our implementation and experimental results of the AV-GHT algorithm. Finally, we have concluding discussions and discussions on future research work in Chapter Six.

CHAPTER 2

SURVEY OF HOUGH TRANSFORM AND ITS GENERALIZATION

The Hough Transform was first introduced by Paul Hough [Houg62] in 1962 for detecting complex patterns of points in binary images. It achieves this by determining the values of parameters which characterize these patterns. The HT is simply a special type of dual transformation technique used in searching for pattern matches in image analysis. Extended patterns are transformed so that they produce certain compact features in a space of possible parameter values. A search for the compact feature in the parameter space can be conducted more effectively and more efficiently than by template matching in image space [Ball81, Davi87a].

This chapter is organized into three sections. We introduce the Hough transform in the first section and give a brief review of the development of the HT for detecting analytical shapes. In the second section, we describe Ballard's generalized Hough transform [Ball81] and review its applications. In section three, we show some architectures used for parallel Hough transform computation.

2.1 The Hough Transform for Analytical Shapes

To illustrate the way that the HT works, we suppose that there is a straight line in an image, and the line consists of a set of collinear points which can be defined by a relation, Π, such that

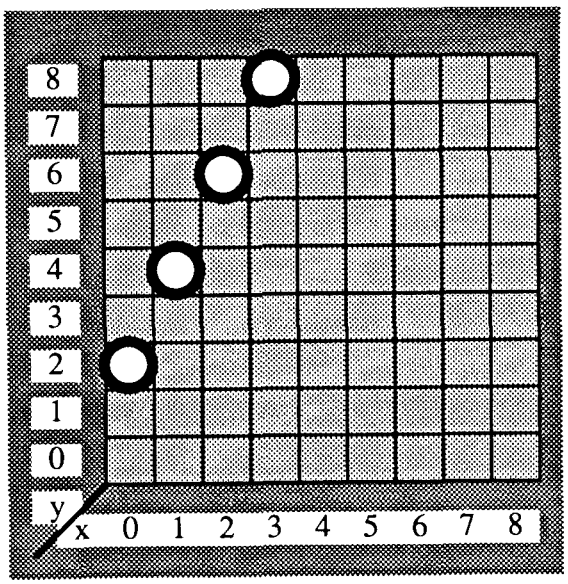
$$\Pi((a, b),(x,y)) = y - ax - b = 0 \qquad (2.1),$$

where a and b are two parameters, the slope and the intercept, which characterize the line. Each parameter pair (a, b) in the parameter space is mapped to a set of points along a straight line in the image space (as shown in the figure 2.1(b)) by relation Π. The backprojection relation,

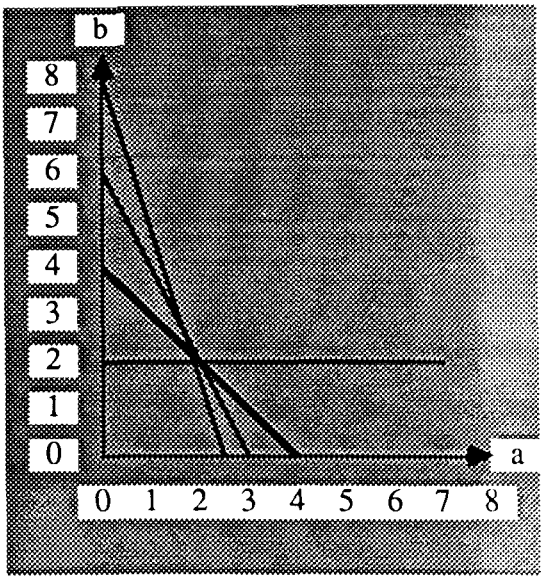
$$\Gamma((x, y),(a, b)) = b + xa - y = 0 \qquad (2.2),$$

maps each image point to a set of parameter values (a, b) which form a straight line in the parameter space. Colinear image points have the same slope-intercept value, so their corresponding lines in the parameter space all intersect at a common point (as shown in Figure 2.1(b)). The coordinates of this point characterize the line in the image space, i.e., they give the value of slope and intercept of the straight line in image space.

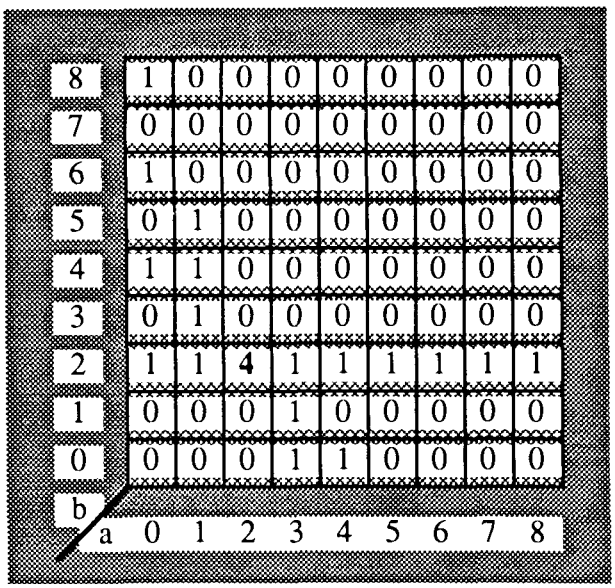
Suppose that an accumulator was set up in the parameter space such that each line will vote into a corresponding entry in the accumulator, and the common point where all the lines intersect will correspond to the peak in the accumulator (as shown in the figure 2.1(c)). Thus HT transforms a global line detection problem in image space into a local peak detection problem in the parameter space. The determination of a local peak should be considerably easier than detecting extended point patterns in image space.



(a) Image with 4 colinear points



(b) Combination of Slope-Intercept



(c) Accumulation Of Votes

Figure 2.1 An Example Of Hough Transform

There have been some variations suggested by several researchers. The most influential one was suggested by Duda and Hart [Duda72] who used a polar coordinate parameterization of straight lines:

$$\rho = x\cos\theta + y\sin\theta \qquad (2.3)$$

where θ is the orientation of the line, and ρ is the distance from the origin of the image to the line. The advantage of the Duda and Hart method is that it uses only a finite amount of parameter space while the slope-intercept parameterization approach has problems handling lines with large slope when the line represented is close to the vertical direction, i.e., for the vertical lines, $b = \infty$. Wallace also suggested a bounded parameterization for lines [Wall85]. His method used the parameters of the two intersecting points of extended line with image boundary. A point on the image boundary is measured by the distance along the boundary counterclockwise from the lower-left corner of the image. Davies suggested that a line can also be parameterized by its point of intersection with a normal vector from the image origin [Davi86a].

It is also very straightforward to extend the HT method of line detection to the detection of analytical curves. Given that a curve is characterized by n parameters, $a_1,...,a_n$, a relation can be similarly defined by

$$\Pi((a_1,...,a_n),(x,y)) = 0. \qquad (2.4)$$

And the backprojection relation can be obtained by swapping the roles of variables (x,y) with parameters $(a_1,...,a_n)$ in Equation 2.4:

$$\Gamma((x,y),(a_1,...,a_n)) = 0. \qquad (2.5)$$

This backprojection relation maps each image point in the image space onto a hypersurface in the n-dimensional parameter space. The common intersecting point of these hypersurface indicates a possible parameter combination of $a_1,...,a_n$.

Kimme, Ballard, and Sklansky [Kimm75] demonstrated the use of HT to detect circular arcs by using edge orientation information to constrain the ranges of parameters. This was the first time that edge orientation information was used in an HT procedure to reduce the computational cost. There have been many papers dealing with detecting other curves by using HT--an example is the detection of ellipses described in Tsuk83 and Tsuj78.

In fact, the Hough transform can be viewed as an evidence-gathering process. Each edge point in an image makes "guesses" on all parameter combinations that could have produced it. Since the continuous parameter space is usually approximated by the union of a number of finite-sized regions, each associated with an entry in an array in a digital computer, such "guesses" are made by incrementing the accumulator entries that correspond to those parameter combinations. The size of the regions of parameters are chosen based on the precision requirements of the parameters.

In the general case where parameter space is multidimensional, the backprojection relation will map an image point to a hypersurface in the multidimensional parameter space. All the hypersurfaces will intersect at a common point if the image contains the shape sought and characterized by the combination of parameters represented by the intersecting

point in the parameter space. If all the entries corresponding to the regions that hypersurfaces pass are incremented, there will be a peak in the entry that corresponds to the intersecting point of hypersurfaces. Therefore, locating the peak in an accumulator array can be used to approximate the computation for the intersecting point of hypersurfaces. Peak detection is much easier and more efficient than matching patterns of points in the image domain.

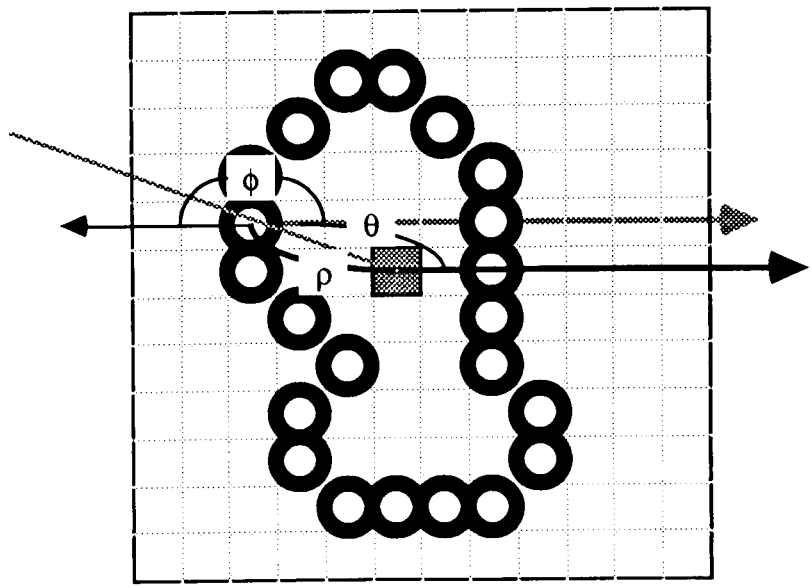
Illingworth and Kittler have recently done a good survey on the HT and its applications [Illi88].

2.2 The Generalized Hough Transform

Attempts to further extend the HT to detect arbitrary shapes have been made by several authors [Illi88]. In 1981, in his well-known paper [Ball81], Ballard extended Merlin and Farbers' work [Merl75] and proposed the generalized Hough transform (GHT) for detecting arbitrary shapes of any orientation or scale [Ball81]. His work obtained considerable computational efficiency by fully exploiting the edge orientation information.

Suppose that we are given a model with a shape that cannot be expressed by analytical equations. In Ballard's GHT, the model was used to construct a so-called R-table. A polar coordinate system is established in the model by fixing a reference point and using it as the origin. The R-table records the positions of boundary points by their polar coordinate values. The rows of the R-table are indexed with the gradient angles of edge points. An example of such R-table construction is shown by Figure 2.2, where Figure

2.2(a) is the model with its polar coordinate system and Figure 2.2(b) shows the R-table for the model in Figure 2.2(a).



(a) Geometry Used To Form The R-table

φ \ (ρ,θ)				
0	(2.2,335)	(2,0)	(2.2,25)	(2.8,45)
45	(3.3,75)	(4,90)	(2.8,45)	(4.2,45)
90	(4.1,100)			
135	(3.6,125)	(3.6,145)		
180	(3.3,160)	(3,180)	(3.6,235)	(4.5,255)
225	(2.2,205)	(2.2,255)		
270	(5.1,265)	(5,270)	(5.1,275)	
315	(5.5,280)	(5,310)		

(b) R-table

Figure 2.2 An Example of R-table Used By The GHT

Suppose that all the objects sought have the same scale and orientation as the model. Because of the way that the R-table was constructed, it is not difficult to imagine that given an edge point (x,y) with gradient orientation ϕ, the possible reference point is constrained at the locations of

$$x_r = x - \rho_\phi \cos\theta_\phi$$
$$y_r = y - \rho_\phi \sin\theta_\phi \,. \qquad (2.6)$$

If a 2D array is used as an accumulator, which is called Hough space or parameter space, and equation (2.6) is applied on all edge points, then the entries that correspond to the reference points will accumulate the highest values (votes). The peak should be equal or close (due to errors) to the number of boundary points of the model since each correct edge point should contribute one vote to its reference point.

In the case where the scale and/or orientation of objects in images are different from the model, different scaling and orientation have to be tested. The orientation and scaling that give the highest peak value determine the size and orientation of the object in the image. It also means that the parameter space will be increased by another 2 dimensions. It certainly will add more computational cost, but compared to template matching that is completely carried out in image space, the GHT is still very efficient.

Davis [Davi82] suggested a hierarchical approach of the GHT. His method is to decompose the complex shapes into simpler shapes. Then the GHT is applied on each part, requiring the detection of compositing parts before detecting the composite. The

shape-recognition process is better controlled and carried out using a top down hierarchical approach. This makes the detection of a partial shape possible.

Similarly, the GHT can also be viewed as an evidence-gathering process. Each image point votes for all the parameter combinations that could have produced it if it was part of the shape sought. The only difference between the GHT and the HT described before is that the GHT references a pre-constructed table (R-table) in its voting while the HT uses analytical functions. The GHT also puts more emphasis on the use of the orientation information of edge points, which further constrains the possible parameter combinations that could have produced the edge point and thus gives the GHT high computational efficiency.

The efficiency of the GHT has attracted many researchers who study its applications under various environments. Adiv applied the GHT to match patterns in two different images resulting from motion [Adiv83]. He proposed to use coarse parameter resolution for large objects, and to partition the image into small sub-images for detecting small objects in order to reduce the computational cost and also to cope with the problem that small objects only produce small peaks in parameter space. He also proposed an iterative focusing approach to parameter estimation.

In their attempt to extend the GHT to 3D [Ball83], Ballard and Sabbath used the Hough technique for detecting groups of parameters sequentially and showed how to derive constraints within the groups.

Kasif et al [Kasi83] have applied the GHT to match subgraphs derived from geographical maps. They compared the GHT with the other methods that could be used for solving the subgraph isomorphism problem and pointed out the possibility of implementing their method on a specially connected parallel machine for parallel processing.

Henderson and Fai [Hend84] have used the GHT to detect objects in 3D data from a laser range-finding system. They reduced the computation for the transform to only a few possible matches by picking some suitable distinctive feature points for the transform. The technique is first to detect planar segments and then find matches with the models. Also, in 3D applications, Silberberg et al [Silb84] used the GHT in estimating the viewing parameters that match straight line segments with the model.

Hakalahti et al studied the GHT for general shape detection [Haka84, Mori85]. They used a two-stage GHT that makes and searches the Hough space from a coarse resolution to a fine one. They also incorporated both contour curvature and local image contrast information into the GHT.

Dhome and Kasvand have used the idea of the GHT to detect 3D polyhedra [Dhom86]. In their method the polyhedra are represented by records containing attributes and relationships between adjacent pairs of planar faces. Images are segmented to construct similar records for candidate planar surfaces. These two record lists are compared to hypothesize all possible locations of objects. Their procedure is hierarchical and employs clustering techniques to identify the hypotheses that occured most frequently.

Davies has published many papers on his studies of the GHT. He has used the GHT to detect both sharp and blunt corners [Davi86b]. He showed that the choice of reference point could have impact on both the accuracy of corner location and the sensitivity for corner detection. He discussed the best compromise of the two factors. In another paper, Davies also investigated the use of GHT to detect polygons of known size [Davi86c]. He proposed that savings on the size accumulator can be made by using the symmetric property. Later he examined the GHT and showed several factors involved in optimizing it [Davi87a]. He suggested that because of sensitivity, each point in the parameter space should be weighted in proportion to the intensity of edge magnitude and in proportion to the *a priori* gradient. He pointed out the difference of object location and sensitivity of object detection and that in order to reduce the computational cost, detection sensitivity is often sacrificed. To enhance the accuracy of the GHT in longitudinally localizing lines, Davies [Davi87b] used ideas similar to the CHough [Brow83]. The CHough is a Hough method proposed by Brown, which allows image edge points to contribute not only positively to the possible parameter combinations but also negatively to the impossible parameter combinations. Davies used two separate Hough spaces to accumulate positive evidence and negative evidence for line segments with known lengths. Line localization is improved by considering the difference between the positive Hough space and the negative Hough space.

Illingworth and Kittler have recently done an extensive survey on the Hough transform [Illi88]. They showed that there were many advantages of the GHT over the

direct template matching and that they could be summarized in the following four major categories:

(1) Random noise in images contributes little to the peaks in Hough space, and therefore it has little effect on the shape detection results.

(2) Because edge points vote independently, partial shape detection can be done by shape model decomposition, and very slightly deformed shapes can be recognized.

(3) Because of the independent evidence gathering, any number of instances of the same shapes can be detected simultaneously in one image frame.

(4) Parallel processing is possible since each edge point votes independently.

Although there are these four major advantages with the GHT, some are only potentially useful. For example, to detect a partial shape a good decomposition of the model into sub-shapes that are capable of matching the partial shape is needed. There are also problems in parallellizing the GHT. For example, the reference point that accumulates the votes for all the points on the shape is the hot spot for causing contentions among PE's and there is no obvious way to solve this.

While the GHT can tolerate slightly deformed shapes, Grimson and Huttenlocher [Grim88] have pointed out that it may also give false results since it is hard to know whether a peak is from a slightly deformed shape or from a similar shape. They also contended that the GHT "will scale poorly, when applied to complex, cluttered scenes, or when using extended features that may be partially occluded."

2.3 Architectures For The HT

Illingworth's extensive survey [Illi88] published late in 1988 showed that almost all the attempts on parallel Hough transform were for straight line detection. Most of them have been restricted to the implementation of the ρ-θ line finding HT. The Hough space is distributed among participating Processing Elements, or PE's, such that each participating PE represents a certain group of ρ-θ combinations. Each PE computes the ρ-θ for its local image point and communicates with the PE that represents this ρ-θ combination. The studies have been focused on determining the best communication routing among PE's that give least contention. Little is known about parallel processing the GHT.

The most commonly used architecture is either the mesh-array or the linear array which are both commercially available. They are running under Single-Instruction Multiple-Data mode and usually each PE is connected to its 2 or 4 or 8 neighbors and has its own local memory. There is a central control unit which decodes instructions and broadcasts to all PE's. All PE's concurrently execute the same instruction on their own data. Some of the architectures require that all of the PE's can only access one location in each PE's local memory at one time (e.g., AIS-5000 [Schm88]). The others provide indirect addressing mode for PE's to access different local memory locations simultaneously (e.g., The new generation AIS [Wils88]). The voting on such mesh-array architectures are usually through synchronized shiftings on all PE's to move each vote for any particular ρ-θ combination to its destination PE.

Little, Blelloch, and Cass [Litt87] proposed a parallel HT implemented on the Connection Machine. The connection machine [Hill85] is one of the first smoothly working prototypes of fully parallel machines. It very closely resembles the parallel machine model of the Shuffle-Exchange Network [Ston71]. It is based on a 12 dimensional hypercube linkage of PE's which allow fast communications between any pair of PE's. Their paper concentrates on resolving the contentions in the voting process but fails to give a performance analysis.

Olsen, Bukys, and Brown [Olse87] studied parallel implementation of the HT on a commercially available system called the "BBN Butterfly Parallel Processor". The BBN Butterfly Parallel Processor [Grow85] is based on a Multi-Instruction Multi-Data (MIMD) architecture and approximates the theoretical machine model called Parallel RAM (PRAM) [Will79]. Each processor in the BBN Butterfly is a Motorola 68000 microprocessor with a substantial amount of local memory (1 MB each). The approximation for a global common memory is achieved by an internal switching network which allows any processor to access rapidly the local memory of any other processor. In their study, Olsen et al achieved a speed-up factor of 80%, i.e., if 100 PE's are used the speed-up is 80 times that of a single PE.

CHAPTER 3

LINEAR GENERALIZED HOUGH TRANSFORM

Detecting a partially-occluded object is a difficult task. Ballard's GHT [Ball81] is a very effective method to detect a complete object. When it comes to detecting partial objects, the straightforward solution of using the GHT is to divide the model into small sub-templates, and apply the GHT to each such sub-template (e.g., Turn85, Davi82). However, this kind of sub-template GHT is limited in precision size to detect a partial object. The relationship among sub-templates varies from one method for division of sub-templates to another. It lacks a systematic approach and may easily cause confusion in real applications.

In this chapter, we first introduce the concepts used in our extension of the GHT and describe a model information extraction process. we start with an object-detection algorithm that best demonstrates our idea. We discuss its merits and weaknesses and then present its robust version called the V-GHT algorithm, followed by an algorithm for detecting all unoccluded parts. However, to simplify the description of our algorithm, we assume that the orientation of objects is fixed[1].

[1] We are not concerned with the rotation and scaling handling in our V-GHT scheme since they can be done by classical methods, i.e., two more dimensions are added into the parameter space. Although these methods are time-consuming, we have not been able to improve upon them.

3.1 Preprocess model

Assume that an image which contains a model of object to be located is given. It is desirable that when the information of the model is extracted individually for each boundary point (along the two dimensions), the relationship between these individual boundary points should also be organized at the same time and in a way that is not very complicated and allows the effective recovery of partial-object information.

To achieve the above two goals, we first set a *reference axis* on the given image model. Although the choice of such a reference axis is arbitrary, we choose a reference axis which cuts through the model object in the image and is parallel to the image border, and we also pick an arbitrary point on the reference axis as a *reference point.* One thing that should be commented upon here is that our choice of reference axis and reference point will not make our following discussion lose generality in any respect. Consider an X-Y coordinate system on an image plane; if there is another coordinate system, called x-y, which can be constructed from X-Y by first moving X-Y's origin to (X_0,Y_0) and then rotating both axes ϕ degree counterclockwise as shown in Figure 3.1, a point P, (x, y), under the new coordinate system, has the following relation to the same point under the old coordinate system (X_1, Y_1):

$$\begin{aligned} X_1 &= X_0 + x\cos\phi - y\sin\phi \\ Y_1 &= Y_0 + y\cos\phi + x\sin\phi \quad . \end{aligned} \qquad (3.1)$$

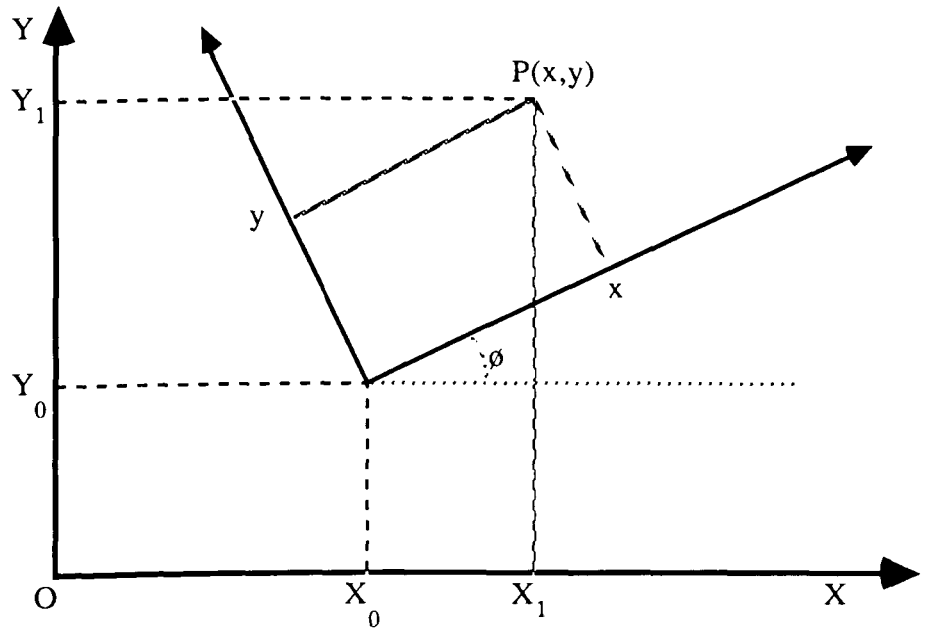


Figure 3.1 Relation between Two Coordinate Systems

By making our reference axis Y-axis and a line which is through the reference point on the Y-axis and perpendicular to the Y-axis as the X-axis, we can convert any other reference choice into our reference system by using the equation (3.1). Such coordinate conversion is independent of the GHT process, i.e., it will not affect the detection in any way except for its required conversion computation.

After the reference axis and reference point are set, we can proceed to extract the information from the given model image with respect to the individual boundary points and the relationship among them. Since we have chosen the reference axis along the vertical direction, unless specified otherwise, we will use the Y-axis and reference axis interchangeably in our following discussion.

3.1.1 Horizontal Information Extraction For Each Point

The first step is to compute a gradient angle for each boundary point. Then, all the gradient angles are sorted, with each angle value only appearing once in the sorted sequence. A *VX-table* is defined as a 2D array whose rows correspond to the sorted gradient angle sequence, one row to each angle, i.e., rows of the VX-table are indexed by the gradient angles of boundary points. Each entry in a specific row of the VX-table is either empty or records a position of a boundary point which has the same gradient angle as the row's indexing angle. The position of a boundary point is represented by the distance from the point to the Y-axis, i.e., its X-coordinate value. The vertical distance from the point to the reference point, i.e., its Y-coordinate value, is not recorded. If more than two edge points from the same image column (same X-coordinate value) have the same gradient angle, only one is recorded into the VX-table. Therefore, the number of valid entries in a VX-table may be less than the number of object boundary points.

So far, the horizontal information for each individual boundary point has been recorded into a table. The vertical information of object boundary points should also be extracted to complete our model pre-processing. We will define a concise vertical relation among object boundary points which can be used effectively and efficiently in the object detection.

3.1.2 V-pattern Construction

The following three properties define a relation called *Vertical pattern (V-pattern)* :

(1) The V-pattern is a numerical array of one dimension,

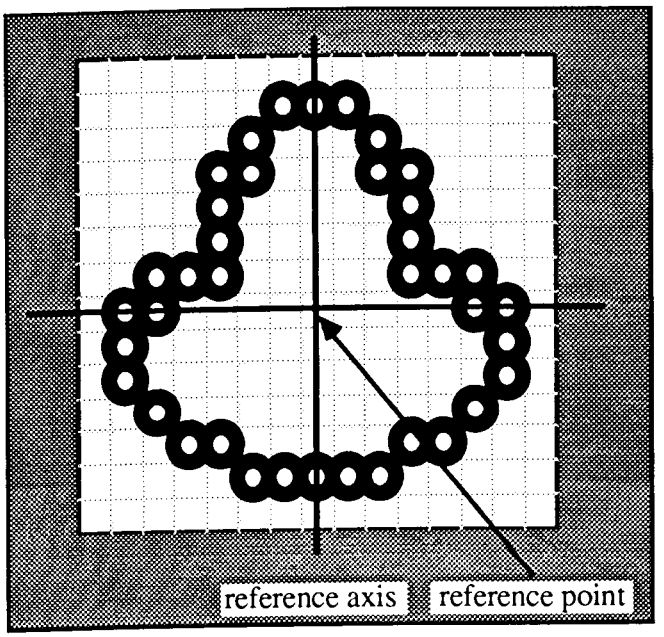
(2) the length of the array is equal to the length of the object perpendicularly projected onto the reference axis, and the V-pattern has the exact correspondence with the segment of the reference axis that the object is projected onto, and

(3) the number of boundary points that are projected onto a single point on the reference axis is the value that is stored in the corresponding entry of the V-pattern.

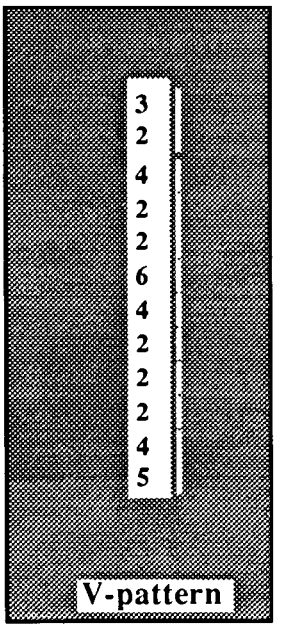
A thus defined V-pattern is in fact a subtotal of object boundary points with respect to each image row and keeps all these subtotals in the order as their row indices; therefore, the V-pattern is a relation along the reference axis of the object boundary points. Figure 3.2 shows an example of an object, its V-pattern, and its VX-table. Now the model information preprocessing has been completed. We are ready to use this information for detections in future input images.

3.2 Detection Algorithm

Our new extension on the GHT is to use a linear pattern to replace the single reference point used in the GHT such that there will be a V-pattern accumulated in the Hough space if the image contains an object sought. In this section, we first give an algorithm called Planar V-pattern Generalized Hough Transform (PV-GHT) to demonstrate how our technique can be used to detect partially-occluded objects. Then we analyze the complexity of the algorithm and discuss its limitations. An example of the comparison of



(a)



(b)

degree index	x						
0	6	3					
45	5	3	2				
90	5	4	1	0	-1	-4	-5
135	-5	-3	-2				
180	-6	-3					
225	-5	-4	-3				
270	-2	-1	0	-1	-2		
315	5	4	3				

(c) **VX-table**

Figure 3.2 An Example of Model Preprocessing

expected accumulation results by the GHT and our Linear GHT technique is illustrated in Figure 3.3. To emphasize the difference between our technique and the GHT, only the contents in the entries in Hough spaces that are of interest are shown.

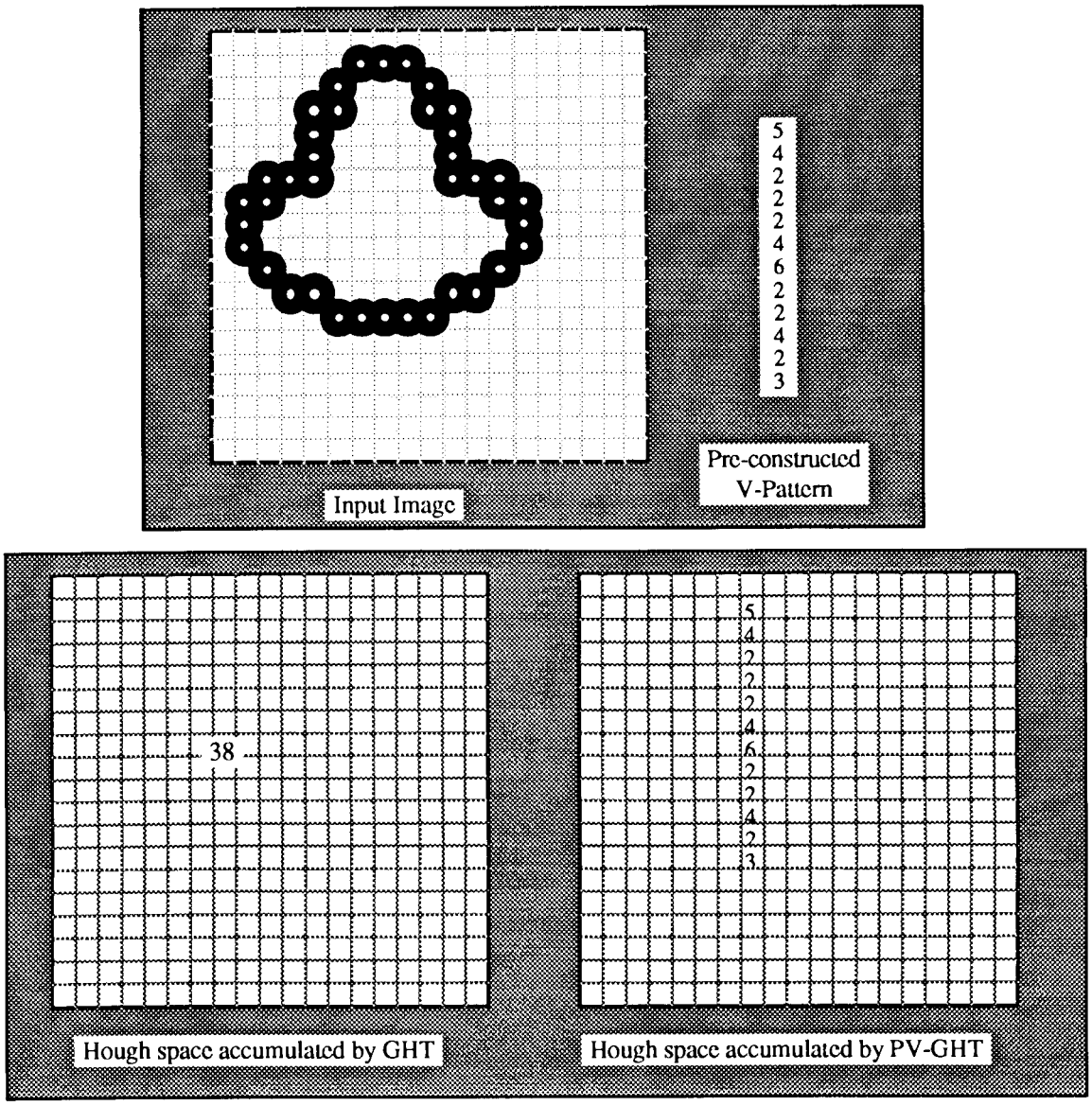


Figure 3.3 A Comparsion of Accumulations In Hough Spaces

3.2.1 The PV-GHT Algorithm

Assuming that both scale and orientation of objects are fixed and the model (VX-table and V-pattern) for the object to be located has been constructed, the algorithm for locating an object can be described as follows:

PV-GHT Algorithm:

(1) Initialize a 2-D Hough space, A, to zero, i.e.,

$$A(x,y) = 0 \quad \text{for all } (x,y).$$

(2) Find all the edge points and compute their gradient angles, and for each edge point (x,y) with gradient angle ϕ do the following:

For each VX-table entry for ϕ, increment the accumulator array

$$\begin{aligned} x_c &= x - x(\phi) \\ y_c &= y \\ A(x_c,y_c) &= A(x_c,y_c) + 1 \,. \end{aligned} \qquad (3.2)$$

(3) For any positive integers i, j and k, if $A(x_i,y_j)$, $A(x_i,y_{j+1})$, ... , $A(x_i,y_{j+L-1})$ has a complete match (or a partial match for a segment) with the V-pattern, then the object sought possibly existed (partially) in the input image. The column x_i should be the reference axis of the (partial) object and the matched segment(s) should indicate the location(s) of those object part(s).

(4) Stop.

3.2.2 Discussions On Match Findings

The final search in the Hough space requires a set of rules to determine whether a segment of the V-pattern is matched to some segments of patterns accumulated in columns of Hough space. This may be trivial for ideal images where there are no distortions and

every object boundary is in perfect shape, although partial occlusions may exist. For such ideal images the final match finding becomes the finding of the longest common substrings between the V-pattern and the columns in the Hough space. The longest common substring problem has been solved [Aho76] and the time complexity is linear to the length of the string searched.

The practical applications often involve processing images that contain slightly deformed object boundaries and noises. Therefore the vote accumulation in the Hough space will not yield a perfect V-pattern even if there is an object sought is in the image. The non-ideal image leads to the need of studying tolerance criteria in match findings. In fact, this kind of problem is not unique to our problem: researchers in pattern recognitions have identified it and have been doing extensive studies in their attempts to solve it. For our case, a substantial statistical study covering a wide range of images might be necessary to produce a set of suitable rules for non-exact matching. However, this thesis will not attempt to solve this problem.

From a logical point of view, a good non-exact match criterion aimed at detecting a non-ideal partial match of V-pattern must give reasonable consideration to the following issues:

(1) Each entry in the V-pattern must be given a reasonable weight such that the entries corresponding to important features are given more weight than the entries corresponding to unimportant features.

(2) There must be certain error tolerance for each entry.

(3) There should be certain tolerance on the number of entries that do not satisfy the standard set by (2).

(4) There should be a limit on minimum length or weight for a partial match.

All of the above four issues should be solved together and consistently: none of them can be solved independent of the others. In our experiment, we took the approach that we deemed to be conservative and appropriate, which is described in Chapter 5 (Implementation and Experimental Results) of this thesis.

3.2.3 Complexities of the PV-GHT Algorithm

The Hough space used in the PV-GHT algorithm is just a two dimensional **NxN** array based on the image size of **NxN**, so the PV-GHT algorithm has $\mathbf{O(N^2)}$ space complexity. Compared with the GHT proposed by Ballard, the PV-GHT algorithm has the same space complexity as the GHT.

The running time differences between the PV-GHT algorithm and the GHT algorithm mainly depend on the matching algorithm. The vote accumulations in PV-GHT algorithm take a less or equal amount of time as the GHT because the VX-table used by the PV-GHT algorithm may contain less valid entries than the R-table (see Chapter 2) used by the GHT algorithm. However, in the worst case, the vote accumulation by the PV-GHT algorithm and the GHT take the same number of steps; the differences are merely locations to vote. For the GHT the voting destinations are the possible reference points, while for the PV-GHT the voting destinations are the possible reference axis. The worst-case time required for the voting step is $O(N^2m)$, where m is the number of columns in the VX-table.

As we discussed before, there can be two types of matching algorithms used in the PV-GHT algorithm. In real industrial applications, it is possible to take images in a fixed environment. If model images and non-model images are taken in such a fixed environment, it is very likely that we could have images wherein the object model in the model image is exactly the same or, at least, if there are any minor differences, they can be fixed by thresholding. In such an ideal match seeking case, we could employ very efficient numeric string matching algorithms presented in Aho76 for a complete V-pattern search in Hough space, and the time complexity is just $\mathbf{O(N^2)}$ for a complete V-pattern search in a NxN Hough space. Under such ideal image assumption, the time complexity of the PV-GHT is exactly the same as the GHT algorithm.

Seeking an exact match is an ideal case. Although it is possible, the majority of real images have certain distortions which cannot easily be fixed, so the matching has to try out all the possibilities which will result in $\mathbf{O(N^2L)}$ running time, where L is the length of V-pattern. Hence, in most practical cases where images are not ideal, the time complexity of the PV-GHT algorithm is $\mathbf{O(N^2L)}$, which is a factor of L slower than the GHT. However, it allows a partially-occluded object to be detected through partial match finding.

3.2.4 Robustness Discussion On The PV-GHT Algorithm

The following example shows the limitation on the robustness of the PV-GHT algorithm. Assume that the alcabash-shape object shown in Figure 3.2(a) is the object sought, Figure 3.4(a) is an input image which contains two overlapping alcabash-shaped objects (their reference axes are also overlapping). If the PV-GHT algorithm is applied to

the image in Figure 3.4(a), the resulting accumulation in the column,which corresponds to the reference axis in the Hough space, is the result of overlapping the corresponding parts of two V-patterns (shown in Figure 3.4(c)). Hence the PV-GHT algorithm will fail to recognize the overlapped V-pattern, so the failure to detect objects will occur with the PV-GHT algorithm.

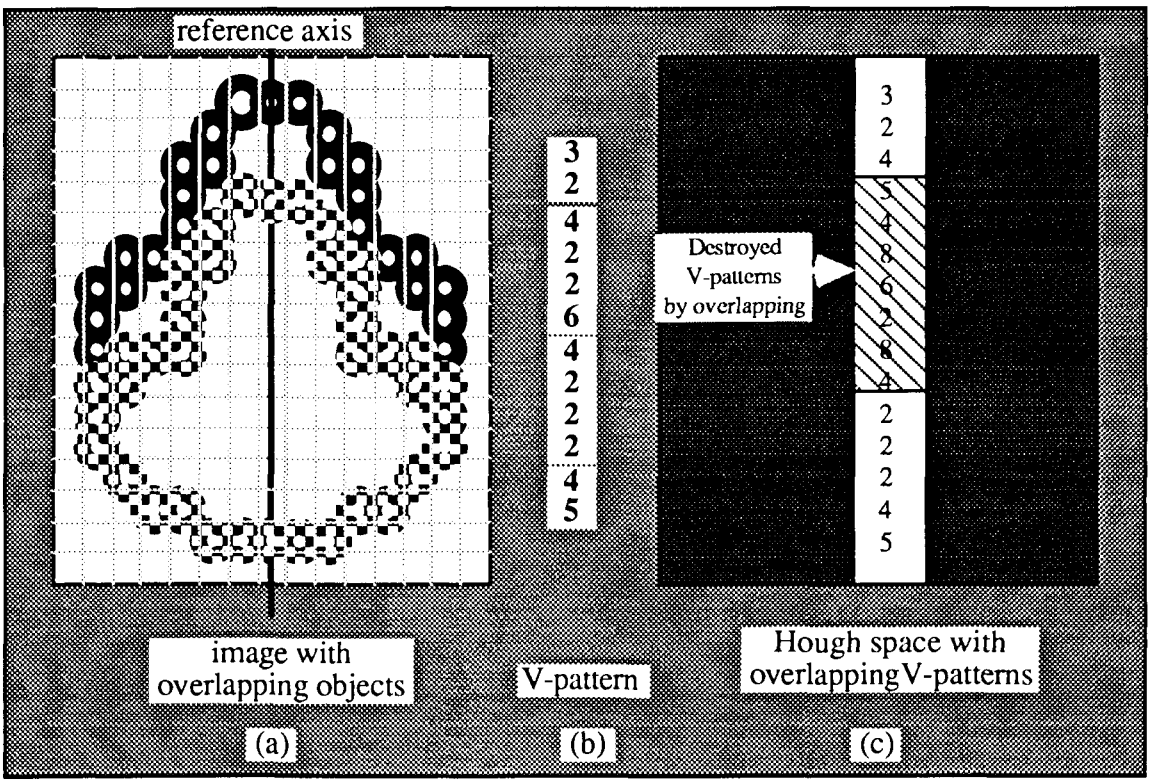


Figure 3.4 Destroyed V-patterns by Overlapping

One might argue that choosing a reference axis other than along the vertical direction for the alcabash-shaped object shown in Figure 3.2(a) will avoid the overlapping of reference axes for the two objects in Figure 3.4(a) and thus make the detection of objects possible. However, when more complicated cases arise, the PV-GHT algorithm may fail again: for example, when an input image has many of the same shaped objects which

overlap one another in every direction. So there is a need to separate every possible V-patterns that would accumulate in the Hough space. We will present more robust algorithms in the following subsections.

3.3 A Robust Linear Generalized Hough Transform Algorithm

As we have seen, one of the major limitations on the PV-GHT algorithm is caused by the overlapping V-patterns in the Hough space. The solution to such overlapping problems is to make the accumulation of V-pattern for each object distinct from each other. To achieve this goal, we combine the reference point technique used in the GHT and the linear pattern technique used in the PV-GHT . The unique reference point for each object is used to make the distinct place for the object's accumulation of V-pattern. In other words, the V-pattern will be accumulated on top of the object's reference point. So the Hough space required to accommodate such V-pattern accumulation becomes three dimensional. Figure 3.5 illustrates this idea. Intuitively, the V-pattern can be considered as "standing" on the reference point.

By such expansion on the Hough space, the overlapping problem shown in Figure 3.4 is solved since the V-patterns for the two overlapping objects will now "stand" on two distinct points, each corresponding to the reference point of an object. The new V-pattern accumulation for the two objects shown in Figure 3.4(a) is shown in Figure 3.6.

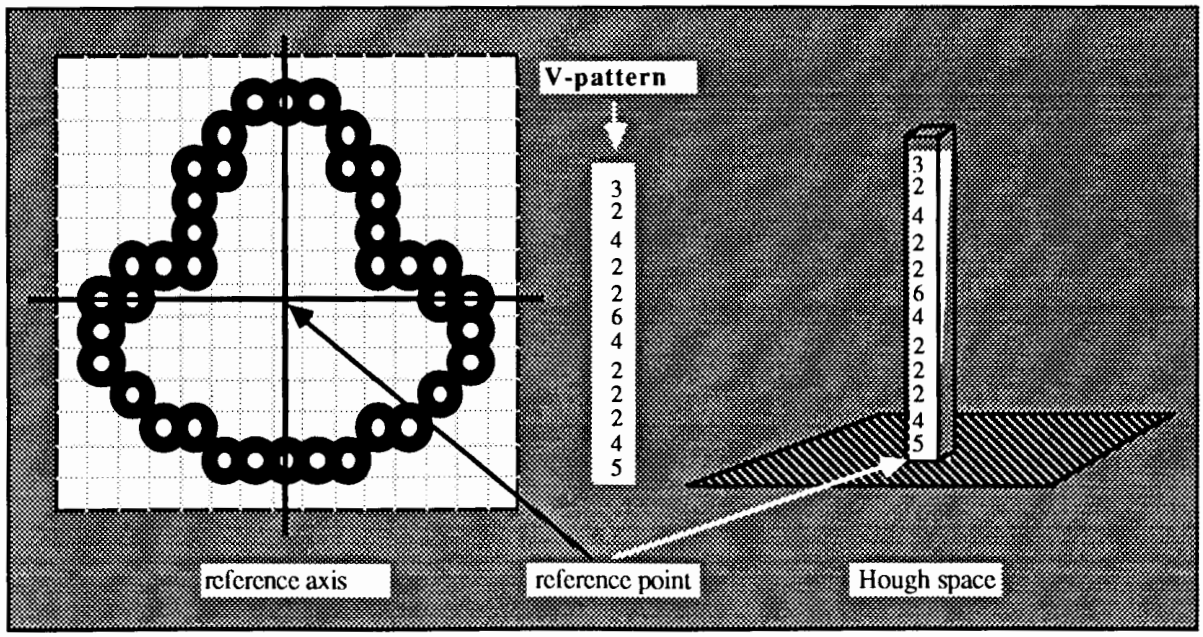


Figure 3.5 V-pattern "Standing" On The Reference Point In Hough Space

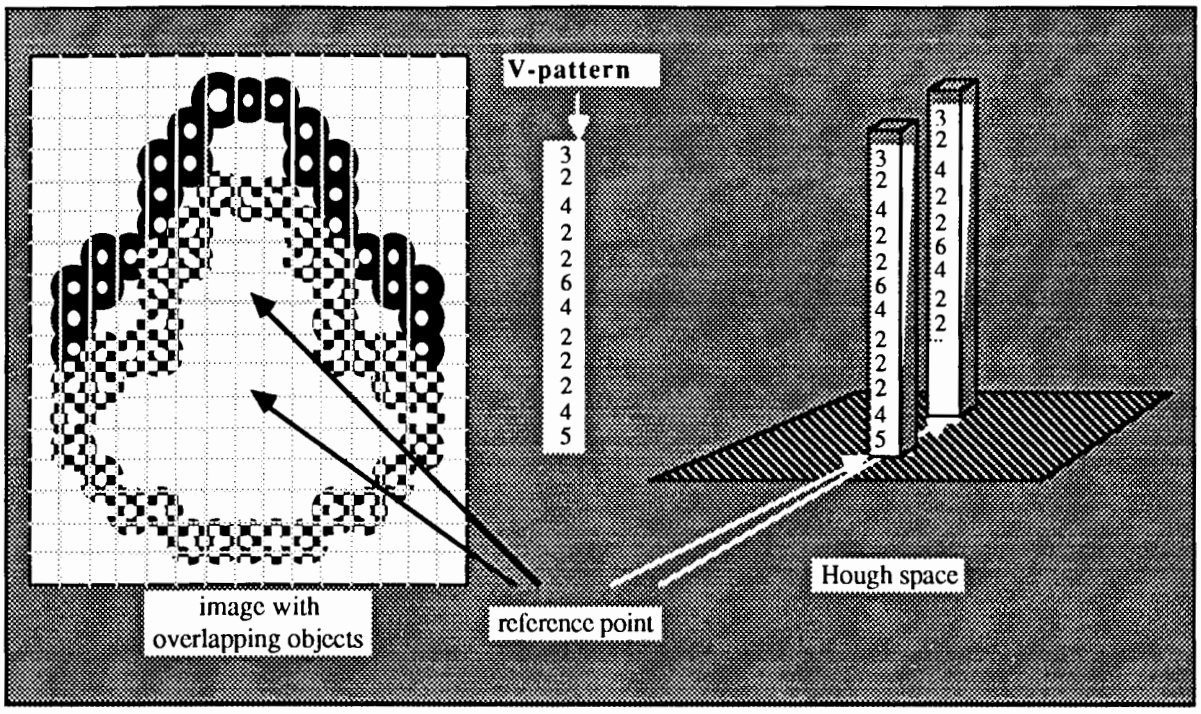


Figure 3.6 Non-overlapping Accumulation of V-patterns

3.3.1 The V-GHT Algorithm

As we intend to make each object correspond to a V-pattern in a unique position (the column on its reference point) in the Hough space, the VX-table has to be expanded so that the y-coordinate information is also recorded. For an edge point to locate its possible reference points, both x-coordinate and y-coordinate values need to be provided by the expanded VX-table. We call this expanded VX-table the *VXY-table*. The number of its valid entries is exactly the same as the number of model boundary points. Figure 3.7 shows the VXY-table for the model in Figure 3.2(a).

degree index	(x,y)						
0	(6, 0)	(6,-1)	(6,-2)	(3, 2)	(3, 3)	(3, 4)	
45	(5, 0)	(3, 1)	(2, 4)	(2, 5)			
90	(5, 1)	(4, 1)	(1, 6)	(0, 6)	(-1, 6)	(-4, 1)	(-5, 1)
135	(-5, 0)	(-3, 1)	(-2, 4)	(-2, 5)			
180	(-6, 0)	(-6,-1)	(-6,-2)	(-3, 2)	(-3, 3)	(-3, 4)	
225	(-5,-3)	(-4,-4)	(-3,-4)				
270	(-2,-5)	(-1,-5)	(0,-5)	(2,-5)	(1,-5)		
315	(5,-3)	(4,-4)	(3,-4)				

Figure 3.7 VXY-table built for Figure 3.2(a)

Assume that the VXY-table and its V-pattern have been pre-computed, and the length of the V-pattern is L. The robust version of the PV-GHT algorithm is called *V-pattern Generalized Hough Transform (V-GHT)* algorithm. The V-GHT can detect a partially-occluded object and is presented below:

V-GHT Algorithm:

(1) Initialize a 3-D Hough space of size NxNxL, A, to zero, i.e.,

$$A(x,y,z) = 0 \quad \text{for all } (x,y,z).$$

(2) Find all the edge points and compute their gradient angles, and for each edge point (x,y) with gradient angle ϕ do the following:

For each VX-table entry for ϕ, increment the accumulator array

$$\begin{aligned} x_c &= x - x(\phi) \\ y_c &= y - y(\phi) \\ z_c &= y(\phi) \\ A(x_c,y_c,z_c) &= A(x_c,y_c,z_c) + 1 \ . \end{aligned} \qquad (3.3)$$

(3) For any pair (x_i,y_i), if $A(x_i,y_i,0)$, $A(x_i,y_i,1)$, ..., $A(x_i,y_i,L-1)$ is a (partial) match with the V-pattern, then the object sought possibly existed (partially) in the input image and the (x_i,y_i) should be the reference point location of the (partial) object.

The following correctness proof of the algorithm shows that the V-GHT Algorithm will detect the unoccluded parts of an object being sought.

3.3.2 The V-GHT Algorithm Correctness Proof

It is trivial to show that in the V-GHT algorithm, all the boundary points of the object sought will vote to to the column on the reference point, since the way that the VXY-table is constructed clearly indicates that the boundary points will "vote back" to the column on the reference point. However, it is not very clear that all the votes accumulated

in the column on the reference point in the Hough space are contributed only by the image points that belong to the boundary of the object sought.

Lemma 3.1:

An edge point will vote to the column on the reference point only if it is part of the shape sought.

Proof:

By formula (3.3), for a point P, (x_p,y_p), with gradient angle ϕ, the votes accumulated will be on the columns that satisfy

$$(x, y) = (x_p - x(\phi), y_p - y(\phi)) \ . \qquad (3.4)$$

If (x,y) is the reference point (x_r,y_r) then (3.4) becomes

$$(x_p , y_p) = (x_r + x(\phi), y_r + y(\phi)) \ . \qquad (3.5)$$

Since the coordinates of the reference point can be chosen arbitrarily, we choose it as the origin, i.e., (x_r , y_r) is set at (0, 0). The equation (3.5) becomes

$$(x_p , y_p) = (x(\phi), y(\phi)) \qquad (3.6)$$

which means that such point P must be a boundary point of the object sought.

This lemma is to show the purpose that the total votes accumulated in the column on the reference point in the Hough space can never exceed the total number of boundary

points of the object sought, and for non-ideal images, it is always less. An example in Figure 3.8 to demonstrates this.

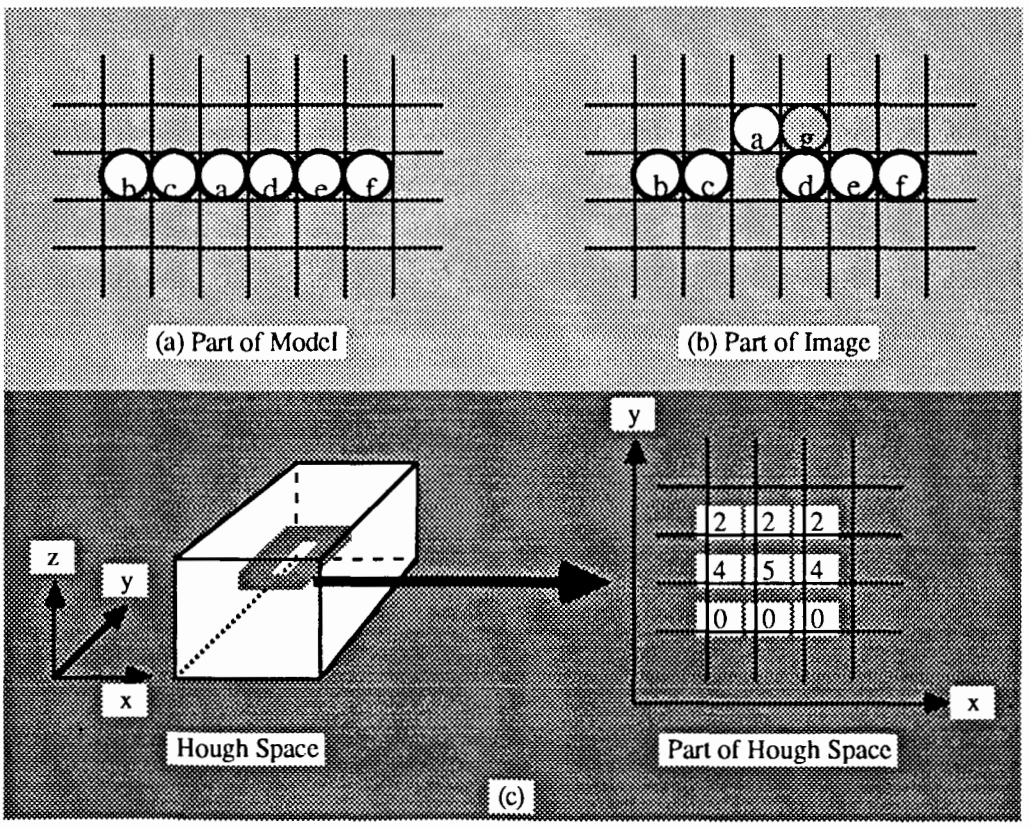


Figure 3.8 Example of Non-ideal Image

In Figure 3.8, part of the image shown in Figure 3.8(b) is somehow deformed from the corresponding part of its model shown in Figure 3.8(a), where point a is shifted by one pixel due to the deformation and point g is a noisy spot. Figure 3.8(c) is the Hough space and the part of one slice which is of interest is enlarged and depicted explicitly. The entry with the number 5 is the one that lost one vote from point a, due to the distortion in the column on the reference point. The entries beside it accumulated votes contributed both by object boundary and noisy spot. This example shows that when the images to be

processed are non-ideal the approximation in the match findings is necessary. The approximation must consider each entry and its neighboring entries together to compensate possible vote loss due to distortions.

Because the V-pattern can be considered to be constructed by counting boundary points in each image row and lining up the results, and the object-detection technique is to find matches in the Hough space to the V-pattern, it is important to show that boundary points of the object sought will vote to the same entry in the Hough space if they are in the same image row. So we give the following lemma:

Lemma 3.2:

Entries in the column on the reference point have one to one correspondence with image rows.

Proof:

For an entry $A(x_r,y_r,z_i)$ in the column on the reference point (x_r,y_r), any boundary point (x,y) voted to it should satisfy the condition that

$$y = y_r + z_i$$

which means the object boundary points that voted to the same entry in the column on the reference point must be from the same image row and the object boundary points in the same row contribute to one particular entry in the column on the reference point.

This lemma implies that the whole Hough space can be horizontally "sliced" into L planes as shown in Figure 3.9: where L is the length of the V-pattern, all entries on each "slice" should only be checked for matching a particular entry in the V-pattern but none of the other entries in the V-pattern. This property facilitates finding a good matching algorithm especially for non-ideal images. It fixes the pairs to be checked in the matching step. If an entry in the i-th plane of the Hough space fails to match the i-th entry in the V-pattern, there is no need to try to match any other entries in the V-pattern.

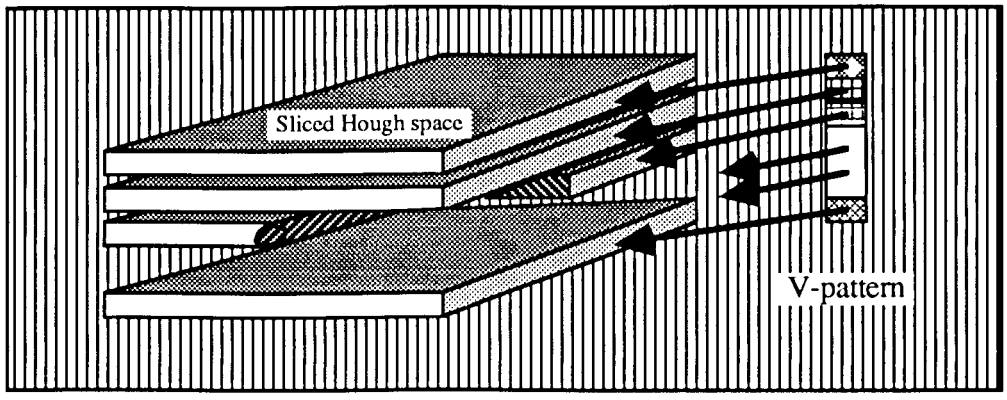


Figure 3.9 Planes in the Hough Space have One-To-One Correspondence with the Entries in the V-pattern

Theorem 3.1:

The V-GHT algorithm will detect a partial object if a partial V-pattern is found in the Hough space by the matching step.

Proof:

By Lemma 3.2, there is one-to-one mapping between image rows and entries in the column on the reference point. The absence of any part of the boundary curve from one

row should not affect the vote accumulation in the segment of entries in the column on the reference point that corresponds to the curve of other rows. ❦

By Theorem 3.1, we show that the V-GHT algorithm can be used to detect a partially-occluded object by finding the longest common substring between the V-pattern and the column on the reference point. The unoccluded case is only a special case of the occluded ones. However, when the end points of an unoccluded part are not in the same row, simply applying the V-GHT algorithm once to the input image will not find the maximum unoccluded part. (An example of this is shown in Figure 3.10.) In fact, several iterations of applying the V-GHT algorithm are needed, with each iteration using a different rotation of both the model image and the input image to recover all the unoccluded parts of the object. This will be discussed later in this chapter.

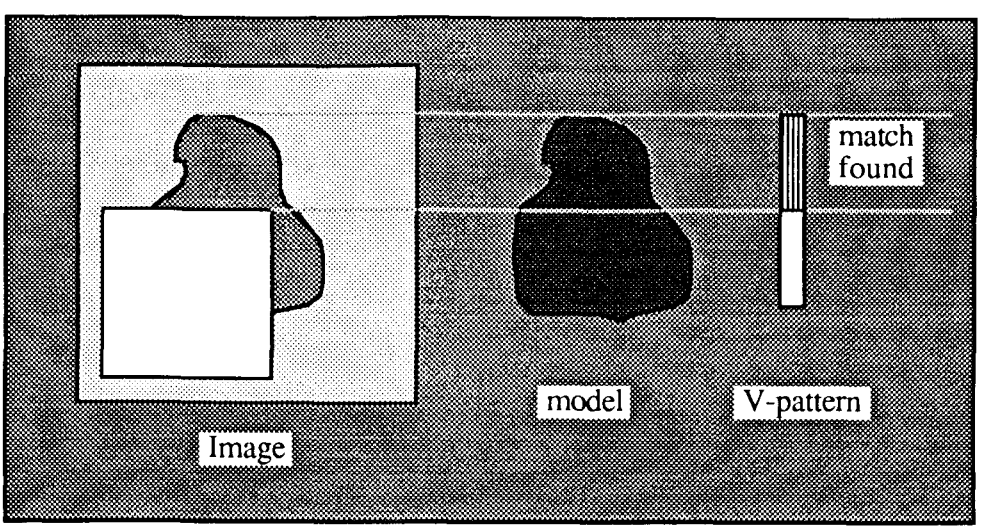


Figure 3.10 An Example of the Unoccluded Part Found

3.3.3 Complexity Analysis

Assume that the size of the input image is NxN size, the VXY-table length is m, and the V-pattern length is L:

- Step one in the V-GHT algorithm takes $\mathbf{O(N^2L)}$ since the Hough space required was $\mathbf{N^2L}$.
- Since there were at most $\mathbf{O(N^2)}$ edge points voting and each of which referred to at most $\mathbf{O(m)}$ entries in the VXY-table, the step two worst case running time is $\mathbf{O(N^2m)}$.
- In the matching step, regardless of what type of matching algorithm or criteria are employed (i.e., an exact matching algorithm for ideal images or approximate matching by some heuristic standard for noisy images), each entry in the Hough space must be examined at least once. But in the V-GHT algorithm, there is no need to examine an entry twice since by Lemma 3.2, levels in the Hough space have a one-to-one correspondence with entries in the V-pattern. Therefore, step three in the V-GHT algorithm has $\mathbf{O(N^2L)}$ time complexity.

To summarize the above analysis, the time complexity of the V-GHT algorithm is $\mathbf{O(N^2(m+L))}$.

In the GHT [Ball81], the voting process has the same time complexity as the voting process in the V-GHT algorithm (step two). In fact, in the voting process the V-GHT does nothing more than the GHT but spreads the votes along a column on the point in the 3D Hough space, which otherwise would fall onto one point in the 2D Hough space with the GHT algorithm. Therefore, the worst case time complexity for the GHT is also $\mathbf{O(N^2m)}$. The V-GHT algorithm has the same order of worst case time complexity as the GHT

algorithm. But its real run time is a little longer than that of the GHT algorithm since it looks for matches in the 3D Hough space while the GHT searches a 2D Hough space for peaks.

The space complexity for the V-GHT algorithm is $\mathbf{O(N^2L)}$, which is a factor of L higher than the GHT or PV-GHT.

3.4 Recovering the Maximum Unoccluded Parts

As discussed before, a partial V-pattern found in the search of Hough space by V-GHT algorithm does not necessarily indicate the piece of object found is the maximum unoccluded part. To find the maximum unoccluded part, a possible approach is to rotate both the model and the image so that the end points of the object part fall into the same row after the rotation. The example in Figure 3.11 demonstrates this idea, where object O, which is partially occluded by an object B, is an occurrence of its model object M. From a partially matched V-pattern of vertical direction, we can find only the arc a->b on object O (shown by Figure 3.11(a)). To recover the arc a->b->c, which is the whole unoccluded boundary of object O, the reference axis should be set in the direction which is perpendicular to a straight line through point a and c, as shown by Figure 3.11(b). Since there is no way of knowing the position of point a and c before we choose a reference axis and apply the V-GHT algorithm to the image, several reference axes of different directions have to be tried before the right one can be picked. In fact, it is not important to find the reference axis that is along the direction perpendicular to line through point a and c, if we can use one reference axis to find some part of arc a->b->c and use some other reference

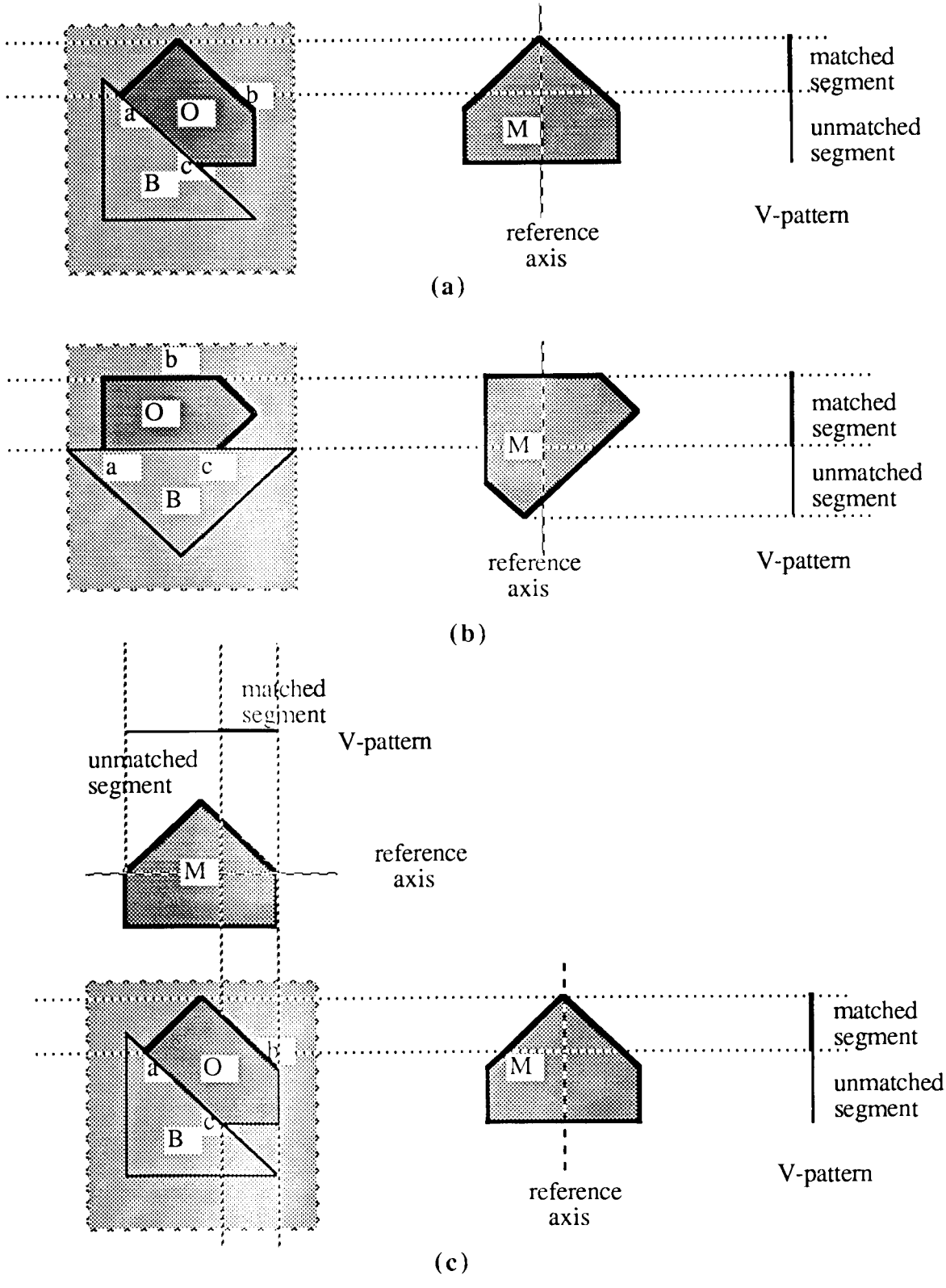


Figure 3.11 Finding Maximum Unoccluded Part

axes to find the rest of the part of arc a->b->c (as shown by Figure 3.11(c)), then we can achieve our goal of recovering the whole unoccluded part of object O by merging all the partial solutions together.

The safest way to recover all the possible unoccluded pieces is to try every possible directional reference axis and put together the pieces found by each trial. However, such trials for all possible directional reference axes is not necessary. For most images, the trials on the reference axes in the x and y directions will suffice. Figure 3.11(c) is such an example. Usually the trials can be limited to the reference axes in as few as 2 to 4 directions.

The trial on each directional reference axis produces a partial result which is recorded in a binary matrix. The final result is achieved by merging all the partial solutions together through an "OR" operation. An example of such a merging procedure for the case shown in Figure 3.11(c) is depicted in Figure 3.12.

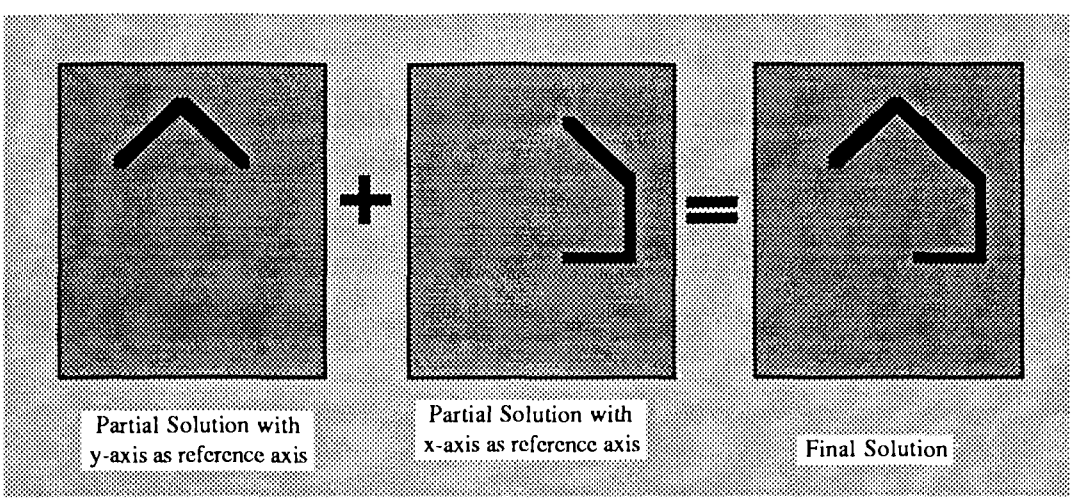


Figure 3.12 Final Result by "OR" Operation on all Partial Solutions

If we still use the same reference point on the model object, the use of axes in different directions other than the vertical one which the VXY-table was built for, will not make the reformation of VXY-table necessary. Each boundary point will still vote into the column on the reference point and it will not matter in which direction the reference axis is set, i.e., for a boundary point (x,y) with gradient angle ϕ, $(x_c, y_c) = (x - x(\phi), y - y(\phi))$ will still indicate the possible reference point. However, different reference axes correspond to different V-patterns and, therefore, different distribution of votes in the column on the reference point in the Hough space, since each V-pattern is the result of the projection of the model object onto a particular reference axis. Given a VXY-table, a V-pattern that corresponds to a reference axis obtained by rotating the vertical reference axis an angle of θ counterclockwise can be computed by the following procedure:

Procedure V-pattern_Construction(θ)

Initialize a one-dimensional array, V, to zero.

For every valid entry (x,y) in the VXY-table do

$$V[x\sin\theta+y\cos\theta] = V[x\sin\theta+y\cos\theta] + 1.$$

/* The array V contains the result of the new V-pattern for direction θ */

End_Of_Procedure.

We can determine the vote distribution along the column on the reference point in the Hough space by method similar to as the computing of the V-pattern described above. Assuming that both the VXY-table and all the V-patterns in all directions, by an angular increment of τ, have been pre-computed, the All directional V-pattern Generalized Hough

Transform (AV-GHT) algorithm, which recovers all the maximum partial pieces of an object sought, can be described as follows:

AV-GHT Algorithm:

(1) Initialize a 3-D Hough space, A, to zero, i.e.,

$$A(x,y,z) = 0 \quad \text{for all } (x,y,z).$$

(2) Initialize a binary image frame F to zero.

(3) Find all the edge points in the input image and compute their gradient angles.

(4) For i = 0 to 180/τ-1 do

$$\theta = i * \tau;$$

for each edge point (x,y) with gradient angle ϕ do the following:

For each VXY-table entry for ϕ, increment the accumulator array

$$\begin{aligned} x_c &= x - x(\phi) \\ y_c &= y - y(\phi) \\ z_c &= x_c sin\theta + y_c cos\theta \\ A(x_c,y_c,z_c) &= A(x_c,y_c,z_c) + 1 \ . \end{aligned} \qquad (3.7)$$

Mark the bits in frame F for any corresponding partial match found in the Hough space with the V-pattern corresponding to the reference axis that was obtained by rotating the vertical axis an angle of θ.

(5) The unoccluded parts of the object sought are indicated in frame F.

Basically, the algorithm runs the accumulation and matching finding procedure employed by the V-GHT algorithm $180/\tau$ times, each time using a different reference axis and V-pattern, and merges all the solutions together.

In most cases, it is not necessary to try every directional V-pattern to pick up every pixel that belongs to the unoccluded boundary part of the object sought. Two V-patterns by a directional difference of 2τ will pick up all the pixels that can be recovered by the V-pattern falling in between them, which suggests that we can even use some larger τ to speed up the process.

3.5 Comparison of the Linear GHT with the GHT

Grimson and Huttenlocher [Grim88] have pointed out that occlusions, noise, and distortion will cause some very serious problems in the GHT algorithm [Ball81] because, in the GHT scheme, the final decision on whether an object existed is totally dependent on the peak value itself. (The peak point would be the reference point of the object sought if it existed.) Compared to the GHT scheme, the V-GHT scheme described in this chapter is more accurate since not only are the votes that contributed to the reference point considered, but the distribution of these votes along the column on the reference point is also tested to decide whether the object sought existed. The V-GHT algorithm can also be used to determine if part of the object sought existed. Such partial object detection can be achieved by the V-GHT algorithm without any modification to the algorithm or any extra computational cost. The V-GHT algorithm handles images with occlusions or without occlusions in the same way. However, such detection accuracy and the partial object

detection achieved by the V-GHT costs moderately more in computational overhead than the GHT.

Another advantage of the V-GHT scheme over the GHT scheme is that it is more suitable for parallel processing. The bottleneck for the parallellization of the GHT scheme is that every boundary point of an object votes to the reference point; therefore the reference point is a hot spot and is difficult for parallel processing. The V-GHT scheme uses a linear pattern to replace the single reference point used in the GHT scheme--so the bottleneck has been loosened up. We will discuss the parallel processing issue further and present parallel algorithms in the next chapter.

CHAPTER 4

PARALLEL PROCESS THE LINEAR GENERALIZED HOUGH TRANSFORM

The real time processing of images is required by some modern application systems and will become an essential requirement in many future image systems. For example, in robotics, the speed to process the images taken by video cameras mounted on mobile robots will critically affect the motion speed of the robot and, in many cases, the usability of such a robot.

Image processing and analysis usually involves a substantial amount of computation. Often the computation consists of some very simple arithmetic operations performed on each pixel in an image frame. It is possible to use a *Single Instruction Multiple Data*, or a *SIMD*, machine to parallelize most of the computations in image processing and analysis. Each pixel in an image usually receives exactly the same set of operation instructions and in the same order as the other pixels in the same image. The instruction sequence is normally done by a "for-loop" covering every pixel in the image frame on a sequential machine. The only differences are the operands which depend on the locations and the pixels themselves.

Parallel computing applied to image processing and computer vision has attracted considerable research attention in recent years [Reev84, Foun86, Cant87]. Many investigations of parallel machine design and application and of theoretically advanced

algorithms have been carried out (e.g., Li90, Reev84, Foun86, Cant87, Duff86, Tani87, Hill85, Batc80, Levi72, Rose83, Dyer81, Duff78).

There have been a number of studies on parallelizing the Hough transform (e.g., [Li90, Cyph87, Rose87, Ibra86]). Illingworth's recent extensive survey [Illi88] showed that almost all the attempts on parallel Hough transforms were for straight line detection. Most of them have been restricted to the implementation of ρ-θ line finding HT. Little is known on parallel processing the Generalized Hough Transform.

Our goal in this chapter is to parallelize the Linear Generalized Hough Transform technique described in the last chapter, and especially to parallelize the V-GHT Algorithm. To analyze the performance improvement of subsequent algorithms presented in this chapter, we will first introduce parallel machine models and then propose algorithms based on them. This chapter is organized into two parts. In the first part, we use a parallel machine model called *Linear Array Processors* (LAP) which has been widely used for image processing. In the second part, we discuss some techniques for resolving voting contentions for potential implementations on a shared memory parallel machine architecture.

4.1 Implementations on the Linear Array Processor

In this section, we will develop and analyze parallel algorithms based on the V-GHT (the V-pattern GHT) scheme that can be implemented on the Linear Array Processors (LAP). The LAP belongs to the class of the mesh array architectures [Reev84].

There has been a substantial amount of work done on this model by many researchers in image processing. A number of machines in the class of mesh array architecture have been built and are commercially available, e.g., DAP [Mark80], CLIP [Duff73], MPP [Pott85], GAPP [Davi85], NON-VON [Ibra86], and AIS [Schm88]. Many practical intermediate-level image analysis algorithms have been implemented on commercially-available machines of mesh array architecture (e.g., Li90, Ibra86, Matt86, Duff85, Pott83). There have been many theoretical advances for mesh computer algorithms (e.g., Nass80, Mill85, Cyph87).

4.1.1 The Model Description

There are some variants in the class of mesh array processor architectures used by researchers to implement their algorithms; we are most interested in those where the Processing Elements (PE's) have the ability to perform *Independent Addressing (IA)* [Fish85, Dani83]. (This will be explained by feature (3) in our following description.)

We are interested in the following four features of a one-dimensional mesh array processor:

(1) A Linear Array Processor consists of p processing elements which are linked into a ring structure. Each PE has a local memory which is sufficiently large to accommodate the memory requirement of the algorithm and a processing unit with the ability to perform typical operations such as reading, writing and arithmetic and Boolean operations. There are permanent identification numbers built in for each PE.

(2) There is a central control unit with links to every PE. It reads instructions from its local memory, decodes them, and then broadcasts the control signals to every PE, such that all the operations are synchronized and the whole system operates in a Single Instruction stream, Multiple Data stream (SIMD) mode.

(3) All the PE's have the ability to perform independent addressing: i.e., rather than all the PE's referencing the same memory location at a given time, they can reference different memory addresses simultaneously. The independent addressing can be realized by using indirect or implied address mode in the instructions issued by the central control unit during the SIMD execution. Each PE, after receiving such instruction, will fetch the content from its dedicated register or the memory location specified in the instruction and use it as the final memory address to be accessed.

(4) There is a common read only memory which can be used to store a table and can be referenced by all PE's simultaneously, and there is an instruction issued by the central unit to every PE to make them reference the data in the corresponding entry of the table concurrently, based on the content in its specified memory location.

Features (1) and (2) described above are the two very basic functions that are supported by all the machines with the mesh array processor architecture. The one-dimensional linear array processor is just a special type of mesh array architecture. The first algorithm presented in this chapter will be based on a linear array architecture that only supports features (1) and (2). So the algorithm will be suitable to be implemented on any machine that is in the class of mesh array architectures.

The second algorithm presented in this chapter will be based on a linear array architecture that supports all four of the features described above. We will show how the

computational efficiency of linear pattern generalized Hough transform is further improved on a more advanced architecture with features of independent addressing and common table lookups. However, the common table lookup function is not necessarily required. It can be easily realized by using the independent addressing function of PE's. Suppose that each PE has retained a copy of the common table in its local memory and has stored the address of the entry in the table which it wants to access in a dedicated register or a special memory cell, then an indirect addressing instruction issued from the central controller will enable all the PE's to access their desired table entries simultaneously. Such common table lookup function can reduce the parallel memory requirement since the whole system retains only one copy of a common table. It will not affect the time complexity of any algorithm.

In fact, the machine that will meet the four specifications will become commercially available in the very near future. For example, the Applied Intelligent Systems Inc., in Ann Arbor, Michigan, USA, is developing a chip called "Centipede" to enhance their existing system AIS-5000 [Wils88]. The AIS-5000 system, manufactured by the AIS company, is currently commercially available and can provide three of the functions described above--the independent addressing is not available. For example, the AIS array processors can simultaneously update an image column based on the contents of a specified common table called the look-up table. Each PE simultaneously copies the content of the i-th entry in the specified common table into its image entry if the number in the image entry was i, (see AISm). Independent addressing will be added into the next generation of AIS system with the new customized PE chip, "Centipede".

Some descriptions of similar machine models which can be found in published parallel image processing papers (e.g., Nass80, Mill85, Cyph87). There is a very detailed model description in Cyph87 which is very close (except in the common table access) to our model, especially in the feature of independent addressing.

In the rest of this section, we will first introduce our algorithm on the LAP architecture without the independent addressing function described by feature (3) in the above model description. Then we will show how the power of independent addressing can further improve the parallelism of PE's.

4.1.2 V-pattern Accumulations in Distributed Hough Space

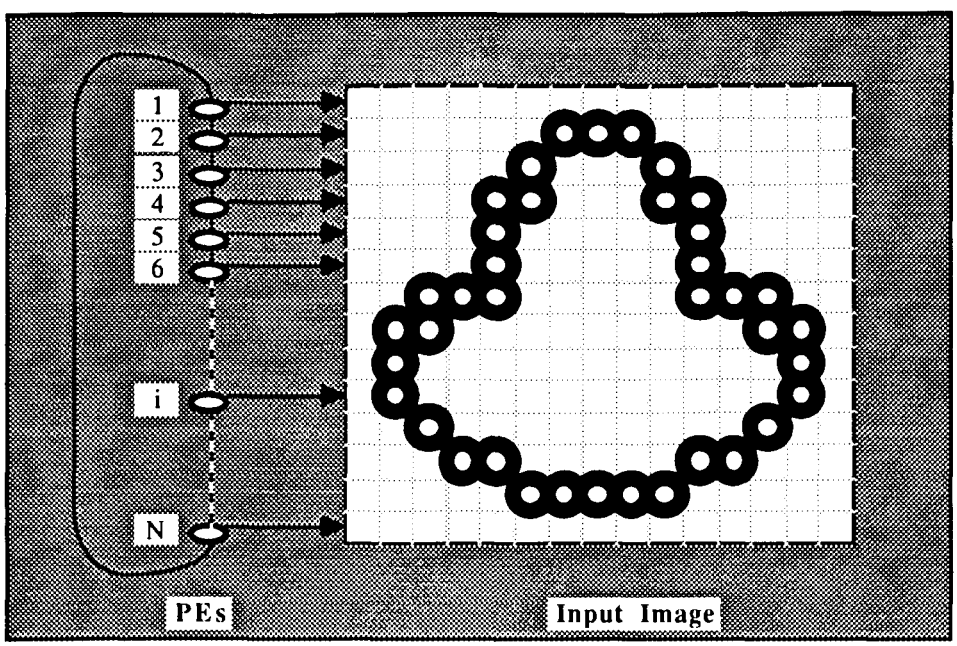


Figure 4.1 N PEs Assigned to an NxN Image

Given a one-dimensional array processor with N PE's which are linked into a ring and an input image frame of size NxN, we adopt the conventional way of assigning PE's to the image, i.e., each PE is assigned to a row in the image frame as shown by Figure 4.1.

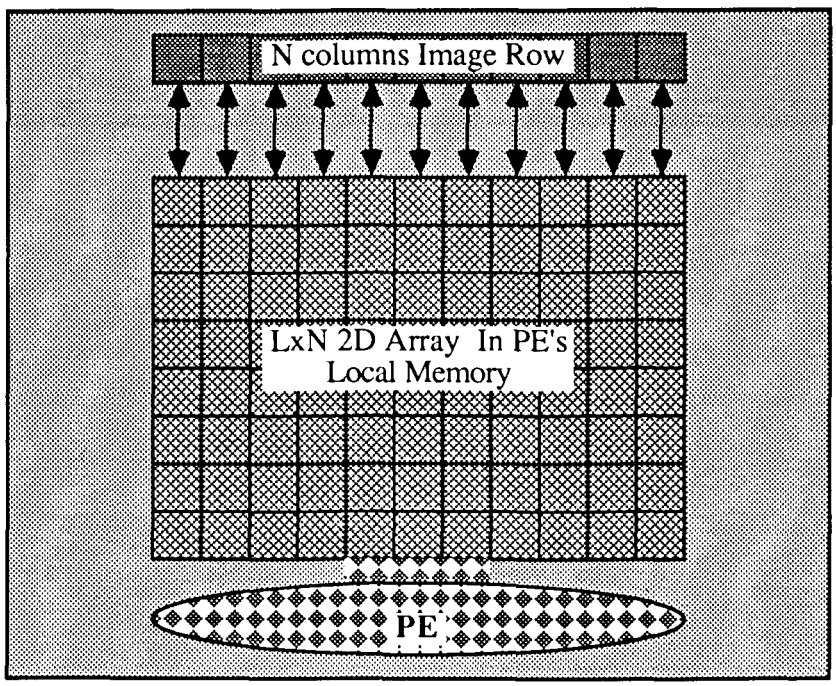


Figure 4.2 A DHS in a PE's Local Memory
with the Corresponding Image Row

The NxNxL 3D Hough space is distributed to N PE's. A 2D array of L rows by N columns in each PE's local memory is allocated. We call such a 2D array *Distributed Hough Space (DHS)*. Each column of a DHS corresponds to an image point as shown in Figure 4.2.

Since there is a total of N PE's, the total number of DHS is N and each of the DHS can be numbered according to the identification number built-in the PE that contains the

DHS in its local memory. By lining up all the N DHS together according to their sequential numbers, a hypothetical NxNxL 3D array (shown in Figure 4.3) is formed which is equivalent to the 3D Hough space used in a V-GHT algorithm.

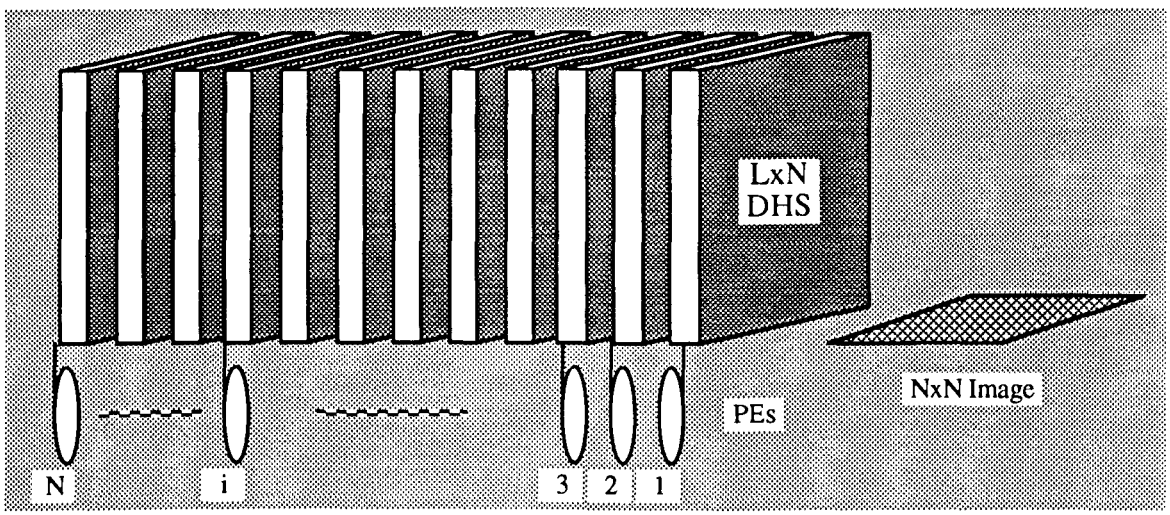


Figure 4.3 3D Hypothetical Hough Space

We define a *Cross Intersect Plane (CIP)* in the hypothetical 3D Hough space to be a plane containing N columns, one column from each of the N DHS, and the N columns corresponding to an image column. An example of a CIP is illustrated in Figure 4.4.

Normally, when implementing a sequential algorithm on an architecture like LAP, there is a large amount of time wasted in communication among PE's. The same thing will happen if the V-GHT algorithm were directly implemented on the LAP machine, since the PE's whose image rows contain some points which are part of the shape sought will vote and communicate with the PE that contains the reference point (the accumulated V-pattern

will be stored in the local memory of this PE). The PE that is in charge of a reference point becomes a hot spot, and every PE that wants to communicate with it has to wait through several steps of "shift". Such shift could require as many as O(N) steps for a vote to reach its destination PE holding the reference point.

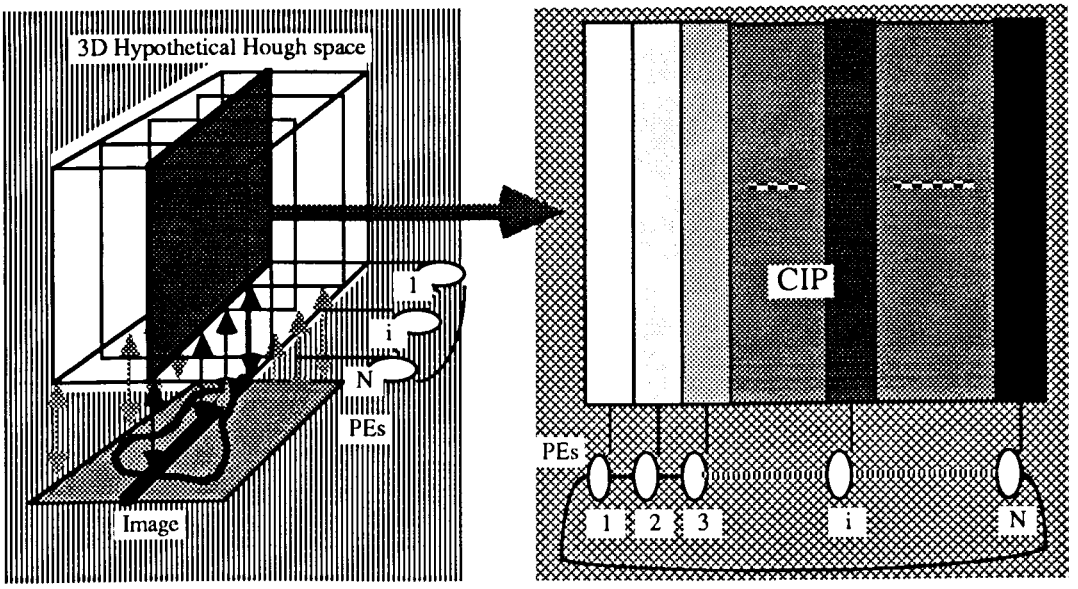


Figure 4.4 An Example of CIP

For example, suppose there is an image edge point in the y-th row and the x-th column voting by reference to a VXY-table entry valued (a,b). According to the equations 3.3, the voting destination is the entry (x-a, y-b,y) in the 3D hypothetical Hough space. Because the image point (x,y) was in the charge of the y-th PE and the voting destination (x-a, y-b, y) was in the DHS of the (y-b)-th PE's local memory, the vote must go through at least the minimum of b and (N-b) PE's to reach its destination. This is shown in Figure 4.5.

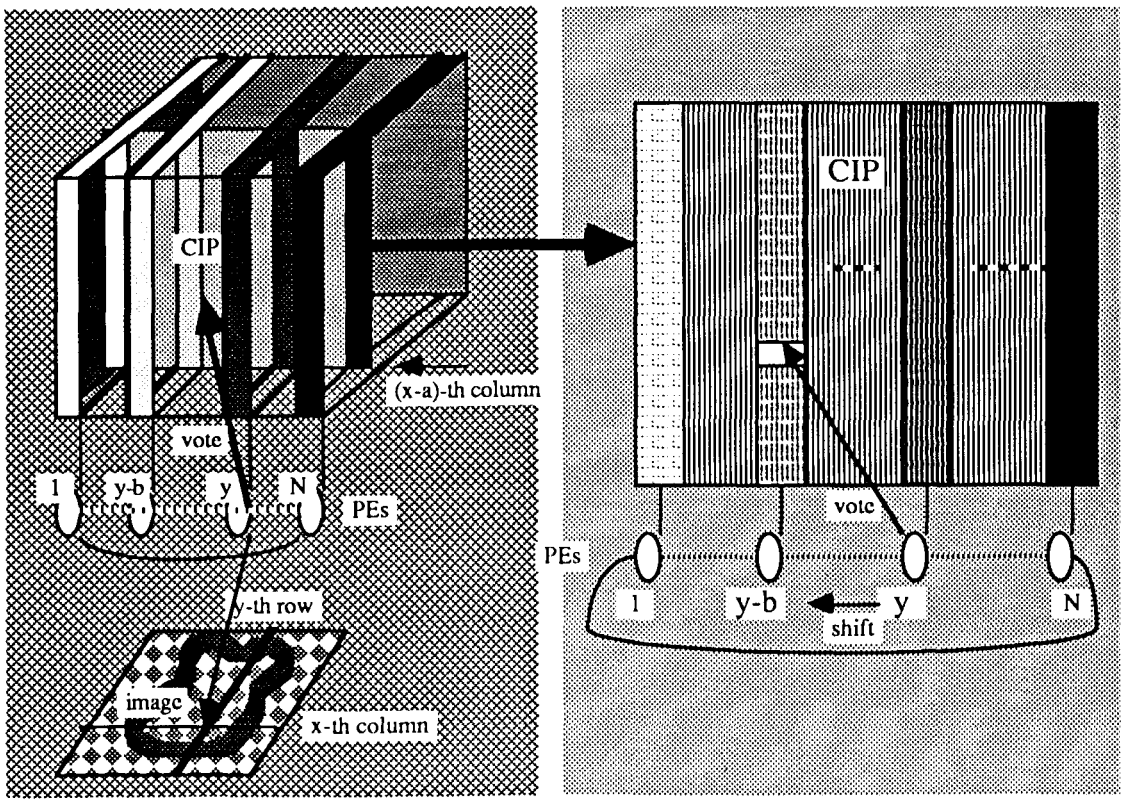


Figure 4.5 An Example of Vote Shifting

To reduce the communication cost among PE's, it is desirable to restrict every PE's voting destination to its local memory as much as possible. This kind of restriction will reduce the communication cost by the voting process. The optimal case would be the one where voting only takes place within each PE's local memory, where the voting result is fast retrievable.

The direct parallel implementation of the V-GHT scheme will allow the entire V-pattern to be accumulated in a single column of a CIP. It will require all the PE's to shift their votes to the PE that is in charge of the column containing the V-pattern. The key that makes no communication possible among PE's during voting is to use a diagonal directional

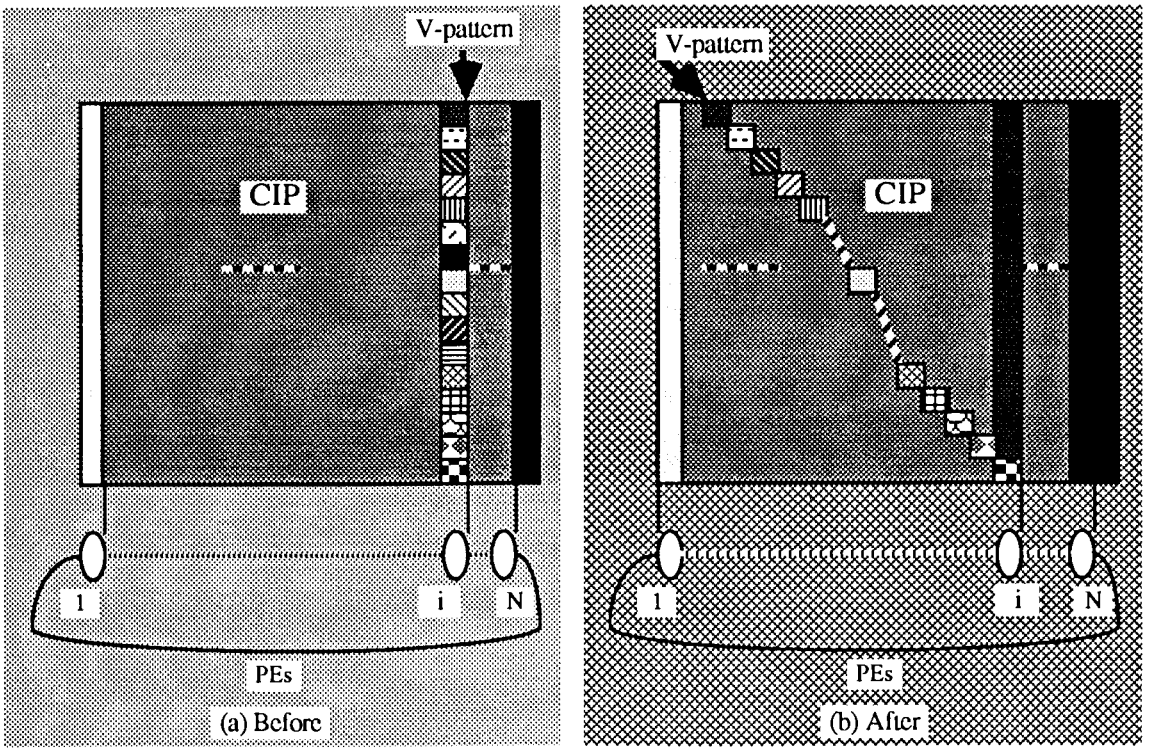


Figure 4.6 Diagonal Directional V-pattern
Compared with the Vertical One

V-pattern seated on the reference point in a CIP instead of the V-pattern in a vertical column on the reference point as shown in Figure 4.6. This kind of diagonal V-pattern accumulation is made possible by the fact stated in the lemma 3.2 that there is a one-to-one mapping between image rows and horizontal planes in the Hough space. The diagonal V-pattern accumulation can be deemed to be the result of the combinations of two synthetic operations: the first one is accumulating a vertical V-pattern in a CIP and the second one is shifting each entry in the V-pattern by the "height" of the entry.

4.1.3 The Algorithm on the Array Processors without Independent Addressing

In this subsection, we will present our first Array Processor GHT (AP-GHT) algorithm which utilized the diagonal V-pattern technique described in the last subsection. The parallel machine model used in this subsection does not require each PE to perform either the independent addressing or common table look-ups. The central controller will look up the VXY-table entries one at a time. So the PE's will only access one, and the same, local memory location at a time to perform voting. This machine architecture is a very basic linear array processor.

We will also discuss the validity of the algorithm and give a formal proof which can be easily extended to prove the subsequent algorithms presented in the later subsections. In fact, the algorithms given in the later subsections are just using the PE's independent addressing function to further improve computational performance.

4.1.3.1 The Algorithm

We assume that the VXY-table and its V-pattern have been pre-constructed with the length of L, and the VXY-table has d rows and m columns. The i-th entry in the j-th row whose index gradient angle is ϕ_j in the VXY-table is denoted by VXY-table$(\phi_j)_i$, and if the entry is not recording a 'nil' value then the recorded coordinate values in the entry are denoted by $x(\phi_j)_i$ and $y(\phi_j)_i$ respectively. We further assume that each of the N PE's has already been loaded with a corresponding image row and its neighboring informations so

that it can determine whether the pixel is an edge point. Given the above assumptions and denotations, the Array Processor GHT (AP-GHT) algorithm that runs on the machine model described in section 4.1.1 (but without the independent addressing function) and parallels the V-GHT scheme described in the last chapter can be described as follows (the variables i, j, x and z are of integer type and serve as iteration counters in the algorithm description):

AP-GHT Algorithm

(1) For x = 1 to N do:

Each PE performs the following operation at the same time:

determine the x-th pixel in its local image row if it is an edge point and compute its gradient direction ϕ if it is an edge point.

(2) For x = 1 to N do:

For i = 1 to m do:

For j = 1 to d do:

Each PE_y (y=1,..., N) that is in charge of an edge point (which is the x-th pixel in its local image row) (x,ϕ_j) performs the following voting procedure at the same time:

(VXY-table$(\phi_j)_i$ is received from the central controller)

if VXY-table$(\phi_j)_i \neq$ 'nil' then

$$x_c = x - x(\phi_j)_i$$
$$z_c = y(\phi_j)_i$$
$$A(x_c,z_c) = A(x_c,z_c) + 1 \qquad (4.1)$$

endif

(3) For x = 1 to N do:

For z = 1 to L do:

Each PE_y (y=1,..., N) compares its local A(x,z) with the corresponding z-th entry in the V-pattern then determines a match or a partial match, and shifts the partial result to its Preceding neighbor PE, i.e., PE_{y-1} (y=2,..., N) or PE_N (y=1).

(4) Stop.

We would like to note here that the equations (4.1) in the step (2) make the voting take place in each PE's local memory, i.e., there are no communications between any PE's involved. The correctness proof of the algorithm will show that after step (2) is completed there should be V-patterns appearing in the Hough space which can be detected by step (3) if the object(s) sought existed.

The capability for each PE to perform independent addressing is not required in the AP-GHT algorithm. The entire contents of the VXY-table are looked up and sent by the central controller to all the PE's, one VXY-table entry at a time, and the PE's vote to just one local memory address at a time based on the VXY-table entry received. In step (3) of the above algorithm the search for V-pattern is carried out in the diagonal direction in CIP's, as shown by Figure 4.7. The search requires all of the N PE's to cooperate such that if a V-pattern is found there must be L consecutively numbered PE's, each compared to and found in just one entry of the V-pattern.

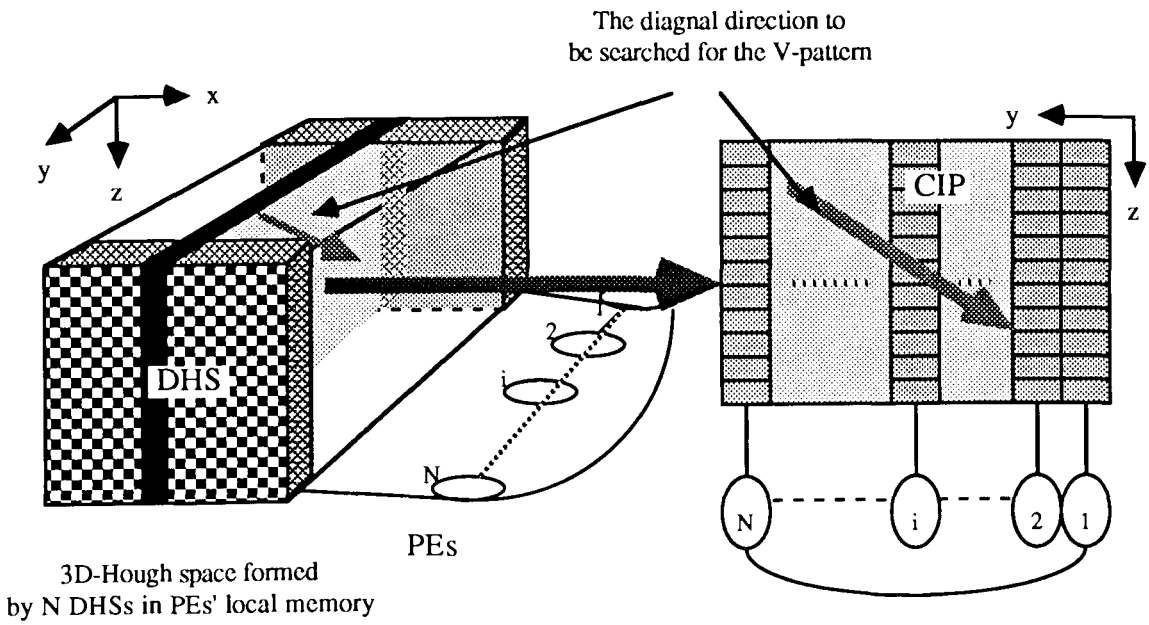


Figure 4.7 V-pattern Searching in the
Hough Space by N PEs

In a practical implementation, the matching information is pieced together by the PE's along the diagonal direction in a CIP. Each PE computes a partial solution which can be symbolically expressed as, e.g., "matched from 1 to 7", then it is passed on to its preceding neighboring PE. During the next iteration the PE receives information from its succeeding neighboring PE, and, during present iteration, will carry on the comparison of its present entry in Hough space with the corresponding V-pattern entry, then put them together to get a new expended partial solution, e.g., "matched from 1 to 8", similar to present iteration. The new expended partial solution will again be passed onto the next preceding neighboring PE. This process continues until the whole V-pattern has been

compared once and then starts the search on the next CIP. There is a total of N CIP's to be searched.

To give a concrete example of V-pattern searching, we associate a binary matrix B of size LxN to each CIP (also size LxN). We will concentrate our following discussions on searching just one CIP since the search of the other N-1 CIP's can be done in the same way. The entry (i,j) in the binary matrix is set to 1 if, and only if, the entry (i,j) in the CIP matches the j-th entry in the V-pattern. An example of the binary matrix is shown in Figure 4.8(a). This binary matrix can be constructed in O(L) steps by N PE's.

The procedure to search the V-pattern is described as follows (z is the counter for iteration, and "first" and "last" are PE's' local variables to record matched segment):

Procedure V-pattern Search (CIP)

Set all PE's (first, last)=(nil, nil).

For z = 1 to L do: (all PE_y, y=1, 2,...,N, do at the same time)

If B(y,z)=1 then if first≠nil then PE_y shift (first, last+1) to $PE_{y-1 \bmod N}$

else PE_y shift (last+1, last+1) to $PE_{y-1 \bmod N}$

else if first=nil then PE_y shift (nil, nil) to $PE_{y-1 \bmod N}$

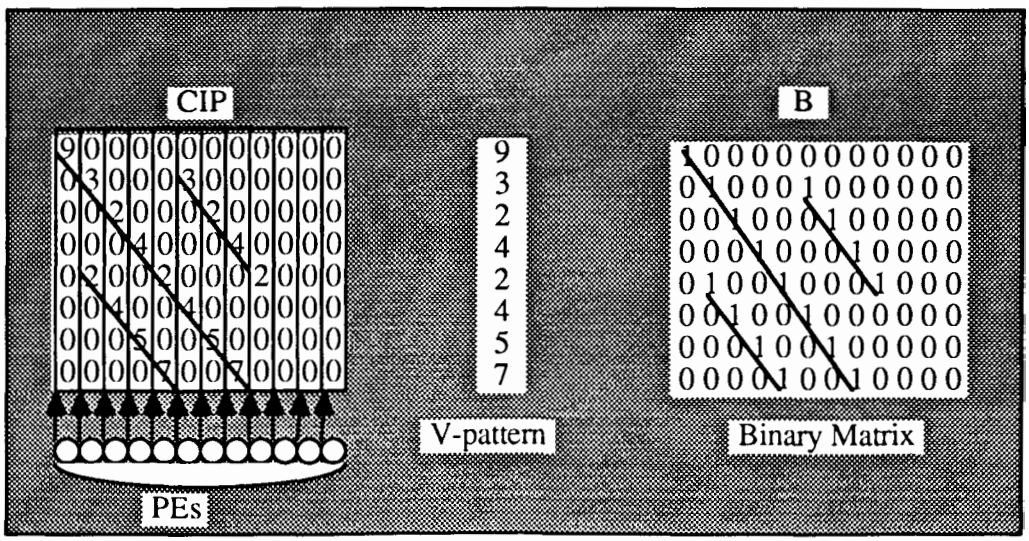
else declare (first, last) a matched segment.

PE_y receives (first, last) from $PE_{y+1 \bmod N}$

EndFor-loop

EndProcedure

Note that the B(y,z) is in the PE_y's local memory. An example of the match searching process is shown in Figure 4.8(b).



(a)

step	Shifting (first,last) among PEs									
	1-->2	2-->3	3-->4	4-->5	5-->6	6-->7	7-->8	8-->9	9-->10	11-->12	12-->1
1	(1, 1)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)
2	(nil,nil)	(1, 2)	(nil,nil)	(nil,nil)	(nil,nil)	(2, 2)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)
3	(nil,nil)	(nil,nil)	(1, 3)	(nil,nil)	(nil,nil)	(nil,nil)	(2, 3)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)
4	(nil,nil)	(nil,nil)	(nil,nil)	(1, 4)	(nil,nil)	(nil,nil)	(nil,nil)	(2, 4)	(nil,nil)	(nil,nil)	(nil,nil)
5	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)	(1, 5)	(nil,nil)	(nil,nil)	(nil,nil)	(2, 5)	(nil,nil)	(nil,nil)
6	(nil,nil)	(6, 6)	(nil,nil)	(nil,nil)	(nil,nil)	(1, 6)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)
7	(nil,nil)	(nil,nil)	(6, 7)	(nil,nil)	(nil,nil)	(nil,nil)	(1, 7)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)
8	(nil,nil)	(nil,nil)	(nil,nil)	(6, 8)	(nil,nil)	(nil,nil)	(nil,nil)	(1, 8)	(nil,nil)	(nil,nil)	(nil,nil)
9	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)	(6, 9)	(nil,nil)	(nil,nil)	(nil,nil)	(1, 9)	(nil,nil)	(nil,nil)

(b)

Figure 4.8 An Example of Match Searching Process

4.1.3.2 Discussions On Validity Of The AP-GHT Algorithm

There are two things that are of particular interest and are also the most important properties for the AP-GHT algorithm:

(1) There were no write conflicts among PE's, and during the voting process, each PE only works in its local memory without communicating with others.

(2) After the voting process a V-pattern will indeed be formed in the diagonal direction in the Hough space if the object sought existed.

The reason that the above two properties are so important is that the second is the goal we wanted to achieve while guaranteeing the first one. In other words, the first property gives the parallel efficiency to the AP-GHT algorithm while the second one guarantees the result is still correct.

Lemma 4.1

There are no write conflicts in the AP-GHT algorithm and no communication between PE's in the voting process.

Proof

There is no common memory to write, and step (2) of the AP-GHT algorithm has each PE perform voting only in its own local memory. #

When using N processing units, the optimal speed-up is by an order of O(N) on the order of processing with only a single PE. One of the major problems for parallel processing is contentions in communication and memory access. Normally, efficiency is

sacrificed for resolving the contention. This lemma shows that the contention problem has been cleanly solved in the AP-GHT algorithm. The next thing that must be shown is that the AP-GHT algorithm did produce the result expected.

To show the AP-GHT algorithm did behave the way expected, we will first do some analysis. The major thing to be shown is that the expected diagonal V-pattern seated on the reference point in a CIP is formed after the voting process in the AP-GHT, algorithm if the image contains the object sought.

Let us first put the N DHS together according to the sequential number of PE's that own the DHS's to form a hypothetical common memory for 3D Hough space. We introduce a dummy dimension y and replace equations (4.1) by

$$
\begin{aligned}
&x_c = x - x(\phi)_i \\
&y_c = y \\
&z_c = y(\phi)_i \\
&A(x_c,y_c,z_c) = A(x_c,y_c,z_c) + 1 \, . \qquad (4.2)
\end{aligned}
$$

It is not difficult to see that such replacement will not create any write conflict among PE's; each PE_y (y=1,..., N) will vote to $A(x_c,y,z_c)$, i.e. the same place in its local memory. So this expended voting equation set does not violate the law of "no communication among PE's during the voting".

Recall that in the lemma 3.2, we proved the property that the object boundary points in any particular row will vote into one and the same entry in the Hough space. Comparing equations (4.2) with equations (3.3), we can see that the only difference is in the y-

coordinate for the Hough space entry to be voted to. But such difference will not change the value in the entry except it is now in a different location in the Hough space because the object boundary points that voted to this entry were all in the same row and referenced entries in the VXY-table with the same $y(\phi)$ value.

Using equations (4.2) to substitute for equations (3.2) in the V-GHT algorithm is just like shifting every entry A(x,y,z) in the accumulated V-pattern by $y(\phi)$ units in the y-axis direction. Since the entry's "height" z is also $y(\phi)$, such synchronized shifting is equivalent to a push to the accumulated V-pattern column into a 45^0 column in a CIP, i.e., diagonal direction as shown by Figure 4.6. At this time, if each PE takes back its own DHS from the conceptual assembled common memory, then it is just what we could have achieved after step (2) in the AP-GHT algorithm. Step (3) will fulfil such 45^0-slope-V-pattern finding.

Formally, we have the following theorem to guarantee the correctness of the AP-GHT algorithm:

Theorem 4.1

The AP-GHT algorithm will detect the object sought if it existed in the input image.

Proof

It is trivial to prove by using the lemma 4.1 and the analysis we just made. ∎

4.1.3.3 The Complexity Of The AP-GHT Algorithm

To show the cost/efficiency of our parallel algorithm, we give the following complexity analysis for the AP-GHT algorithm described in this section (the notation used is the same as in section 3.2.4):

(1) In step (1), inside the for-loop the computation takes constant time, $\mathbf{O(1)}$. Since it takes N iterations to process all the pixels, the complexity of step (1) is $\mathbf{O(N)}$.

(2) It takes $\mathbf{O(md)}$ steps for an edge pixel to finish voting, and sequentially processing a N-pixel row by a single PE will take $\mathbf{O(mdN)}$ time. Therefore, the step (2) has the time complexity of $\mathbf{O(mdN)}$.

(3) Comparing an entry in the Hough space with its corresponding entry in the V-pattern, and then putting the partial solutions together, takes a constant time, i.e., $\mathbf{O(1)}$ time, by a single PE. The two for-loops have time complexity $\mathbf{O(NL)}$. So the total running time for the step (3) is $\mathbf{O(NL)}$.

To summarize the above analysis, the total time complexity of the AP-GHT algorithm is $\mathbf{O(N(md+L))}$ with N PE's on a NxN size image. Compared with the sequential algorithm of V-GHT described in the last chapter, which has $\mathbf{O(N^2(m+L))}$ running time, the AP-GHT algorithm reduced the running time complexity of the V-GHT algorithm by a factor of $\mathbf{N/d}$ with N PE's. If the number of rows in the pre-constructed VXY-table is taken to be a constant, the speed-up achieved by the AP-GHT algorithm can be considered linear.

Normally, the number of rows in a VXY-table is limited. We have:

$$d = 360 / \tau$$

where τ is the increment of angle used to scale the gradient orientations. In real applications, the τ is set between 5 and 20 degrees, so d is a constant and normally lies between 18 and 120. Compared with normal image size of 512x512, i.e., N=512, the speed-up is 4 to 28 times. One must bear in mind that steps 1 and 3 of the AP-GHT algorithm are N times speed-ups. So the speed-up is more than 4 to 28 times, or in other words, the real running time of the AP-GHT algorithm on N PE's is much faster (4 to 28 times) than the V-GHT algorithm running on a single PE.

4.1.4 The Algorithm on the Array Processor with Independent Addressing

We have just shown the parallel computational efficiency achieved on the one-dimensional array processor without requiring the independent addressing function. It shows the parallel capability of our linear Hough transform technique. One of the advantages of our linear pattern GHT over the Ballard's GHT is its efficiency for parallel processing. In this subsection, we discuss how to use the independent addressing function of a linear array processor to reduce the AP-GHT algorithm's running time by a factor of d.

In the AP-GHT algorithm, the central controller sends to all PE's d instructions with d VXY-table entries for d distinct gradient angles, one at a time. Each PE votes only when it receives the VXY-table entry for the same gradient angle as its image point. Therefore each PE's efficiency is less than 1/(d-1) in the AP-GHT algorithm. The efficiency can be improved when the central controller sends only an indirect addressing instruction to all PE's such that all PE's can fetch their desired VXY-table entries and vote

all at once. The following Independent Addressing Array Processor GHT (IAAP-GHT) voting procedure implements this idea.

With the same assumptions made for the AP-GHT algorithm in section 4.1.3.1, the IAAP-GHT voting procedure that runs on the machine with full features (described in section 4.1.1) can be described as follows (the variables i, x and z are integer type serving as iteration counters in the algorithm description):

IAAP-GHT Voting Procedure:

For x = 1 to N do:

For i = 1 to m do:

Each PE_y (y=1,..., N) that is in charge of an edge point (which is the x-th pixel in its local image row) (x,ϕ) do:

if VXY-table$(\phi)_i \neq$ 'nil' then

$$x_c = x - x(\phi)_i$$
$$z_c = y(\phi)_i$$
$$A(x_c,z_c) = A(x_c,z_c) + 1 \qquad (4.3)$$

endif

It is worthwhile to mention that the synchronization among PE's is not mandatory in this IAAP-GHT voting procedure, although the IAAP-GHT voting procedure is designed for a SIMD machine. This means that the IAAP-GHT voting procedure can also be easily implemented on a non-SIMD machine.

The correctness of the IAAP-GHT voting procedure can be proven in a way similar to the proof in section 4.1.3.2.

It takes O(m) steps for an edge pixel to finish voting, and sequentially processing a N-pixel row by a single PE will take O(mN) time. Therefore, the IAAP-GHT voting procedure has the time complexity of O(mN). Compared with the AP-GHT algorithm, the voting complexity is reduced by a factor of d. This is because all of the PE's are allowed to read the VXY-table at the same time and perform voting to different local memory locations at the same time.

The total time complexity of the AP-GHT algorithm is $\mathbf{O(N(m+L))}$ with N PE's on a NxN size image if the IAAP-GHT voting procedure is being used. Compared with the sequential V-GHT algorithm described in the last chapter, which has $\mathbf{O(N^2(m+L))}$ running time, the AP-GHT algorithm with independent addressing voting procedure reduces the running time complexity of the V-GHT algorithm by a factor of N with N PE's, and thus achieves a linear speed-up. In this sense, it is also an optimal parallel implementation of the sequential algorithm V-GHT in that it uses a one-dimensional array N processor.

4.2 Discussions on the Use of a More Powerful Parallel Machine

Advanced theoretical studies have long been proven to play an important role in the creation and evolution of computer systems. The Mesh-connected arrays of processors, which were first proposed more than thirty years ago [Unge58] for parallel image processing, is one such example.

It is worthwhile to examine how our new Linear Pattern Generalized Hough Transform technique could exploit the parallel computational power of a shared memory machine architecture where a virtual global memory exists and is accessible to all PE's.

There is possible contention through concurrent writing to one memory cell by more than one PE at one time. The way to solve such contentions is to arrange the access sequence so that write operations on any one location proceed one by one because of a particular structure in the data organization.

When the VXY-table was introduced in the last chapter, there was no restriction imposed on its organization except that its rows were indexed by gradient angles of the object boundary points. There was no rule on the order in which the data should be recorded within each row and how data elements in different rows should relate to each other. Although this will not matter in a sequential process of images, there are potential contentions in parallel processing caused by using such a VXY-table. To eliminate the potential contentions which result from using a VXY-table, we can re-organize the VXY-table imposing a rule:

> If two entries have the same y-coordinate values then they must be recorded into different columns in the table.

Since the PE's will vote according to equation 4.3, the MP-table makes the PE's' voting destinations distinct from each other at any time. Therefore the contention-free parallel voting by NxN PE's is possible. This will lead to a possible linear speed-up implementation of the V-GHT on such an ideal shared memory parallel machine.

CHAPTER 5

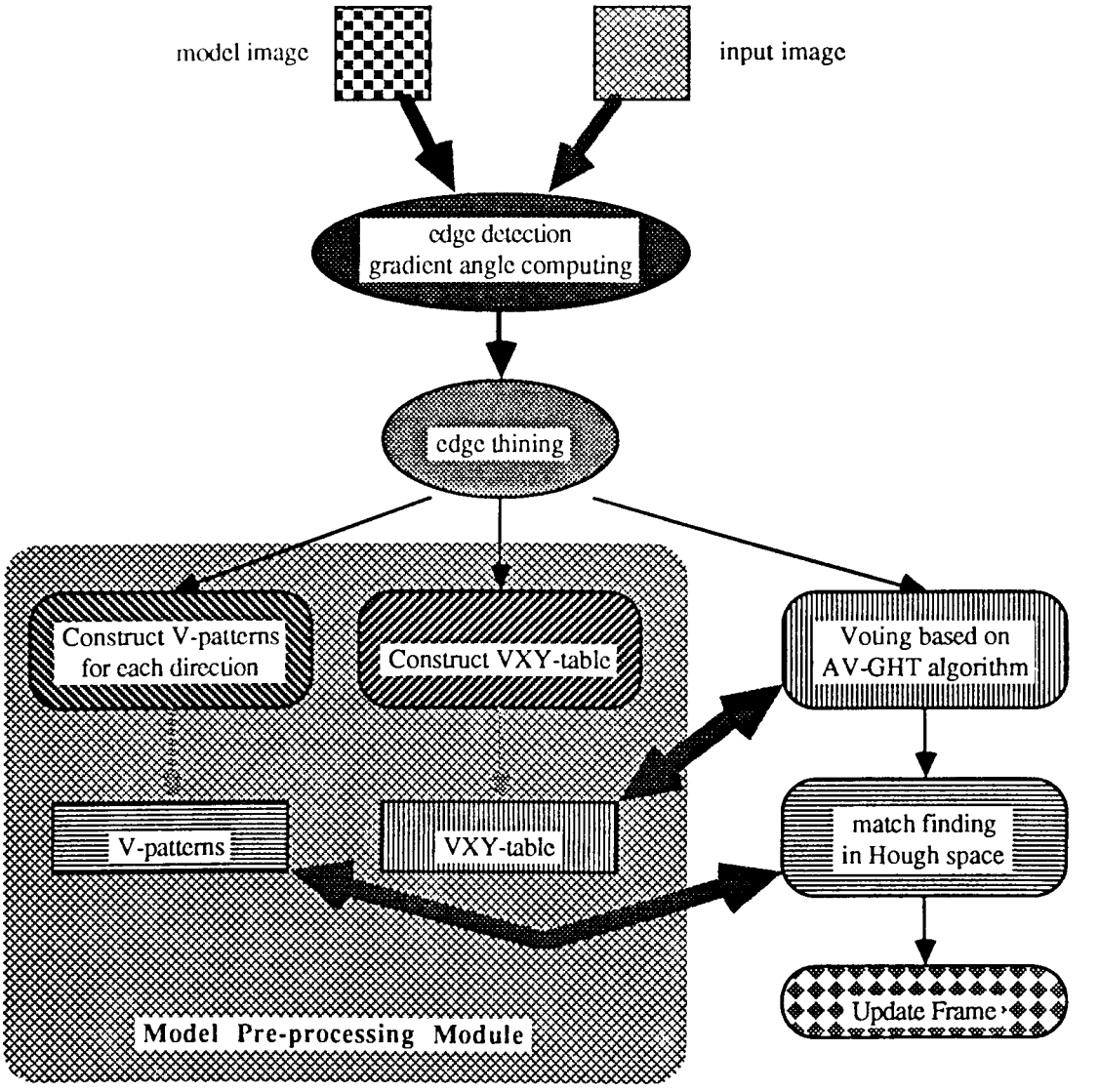
IMPLEMENTATIONS AND EXPERIMENTAL RESULTS

To test the object recognition algorithms developed in the previous chapters, an experimental object recognition system has been constructed and some interesting experiments have been performed on the system.

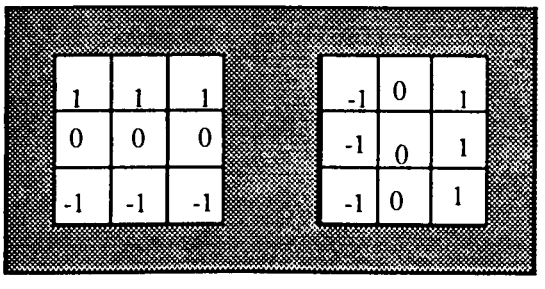
5.1 System Implementation

The core of our object detection system is based on the AV-GHT algorithm described in Chapter 3 which, compared to either the V-GHT algorithm or the PV-GHT algorithm, is capable of dealing with more general images and more robustness in the presence of occlusions or noises. It is implemented in C under UNIX environment and utilizes the International Image System (IIS) for the purpose of digitizing input images and displaying output images.

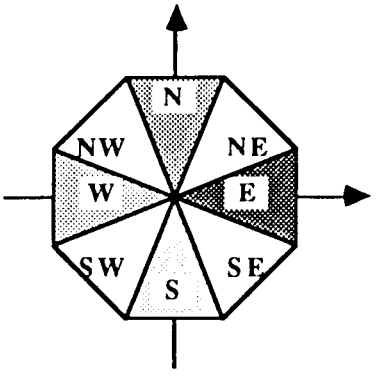
The structure of the system is shown in Figure 5.1(a), where the Edge Detection And Gradient Angle Computing is performed by applying the 3x3 gradient operators shown in Figure 5.1(b). For the model image, one extra step is taken to remove those edge pixels that are not on the boundary of the object. This is done by using grey level information. The 360^0 angle is divided into eight equal sections, i.e., 45^0 for each section as shown in Figure 5.1(c). We define the *directional neighbors* as the two edge pixels whose gradient directions are in the same section and next to each other in such a direction. The Edge Thining is done by applying the following rule:



(a) System Structure



(b) Edge Operators



(c) Angle Division for Thining

Figure 5.1 System Structure

> For any two directional neighboring edge pixels with the gradient angles having a difference of less than half of the section width, i.e., 22.5^0, the one with greater edge magnitude prevails. In the case of two edge pixels with equal magnitudes, the one coming first from the direction suppresses the other.

The constructions of the VXY-table and the V-patterns for the vertical reference axis, its 20^0, 40^0, ..., 160^0 rotations and the votings are as described in Chapter 3. We put equal weight on all entries of V-patterns and implemented the Matching in the Hough Space with the following criteria:

(1) An entry in V-pattern matches its corresponding entry in the Hough space if the two entries differ from each other by less than 25 percent of the value in the V-pattern (tolerating small error on each entry),

(2) contiguous unmatched entry pairs in a matched segment must be less than 5 in length and each unmatched pair must not differ more than 2 or 50 percent of the value in the V-pattern (tolerating a certain amount of moderate error caused by breaks in a segment), and

(3) a matched piece must exceed 15 percent of the length of the corresponding V-pattern (declaring a very small partial match is not meaningful).

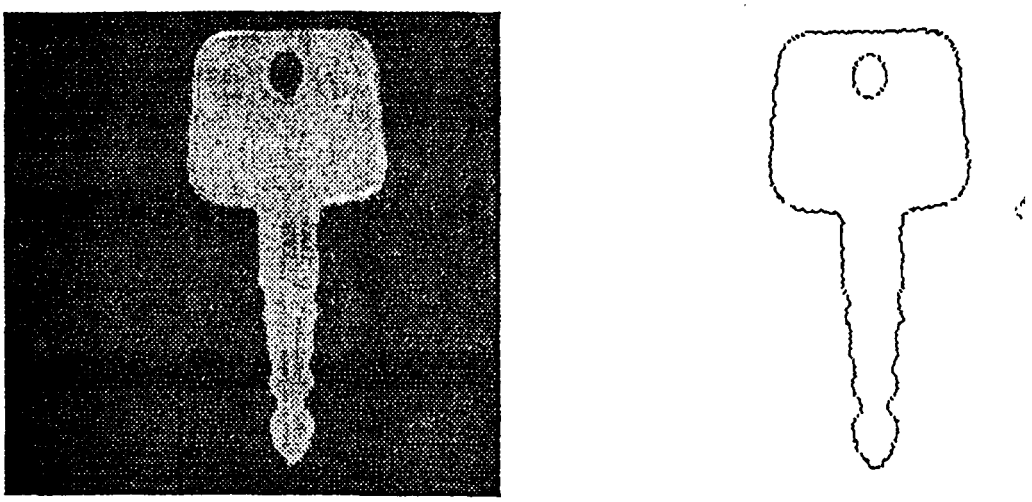
The above criteria are obtained through experiments on several relatively good quality images. The actual numbers chosen for the criteria are totally based on the images used in our experiments and had to be adjusted for each particular set of image data. We deem such a set of criteria to be conservative and appropriate. We take such a relatively

conservative standard because of the lack of statistical studies on those image qualities which cause deviations in the V-pattern accumulation.

We used a NxN binary matrix to record the matched parts and kept updating it each time the match finding completed a rotation of the V-pattern. When all the rotations have been processed the matrix contains all the unoccluded parts.

5.2 Experimental Results

A key is used as our model object for testing (shown in Figure 5.2(a)). The extracted boundary for the key model is shown in Figure 5.2(b). Two images of size 256x256 have been tested, one for the no-occlusion case, the other for the occluding case



(a) (b)

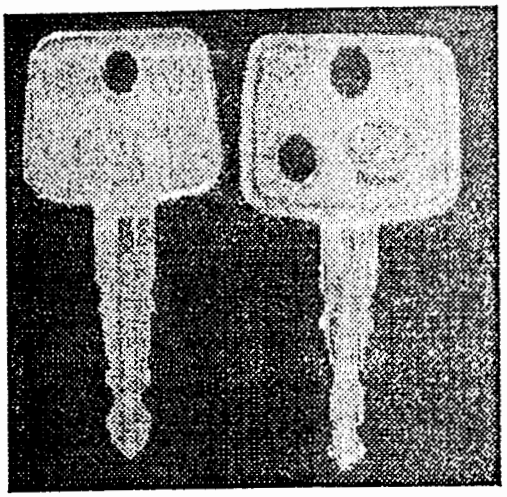
Figure 5.2 Model Key and its Boundary

5.2.1 No-occlusion Case

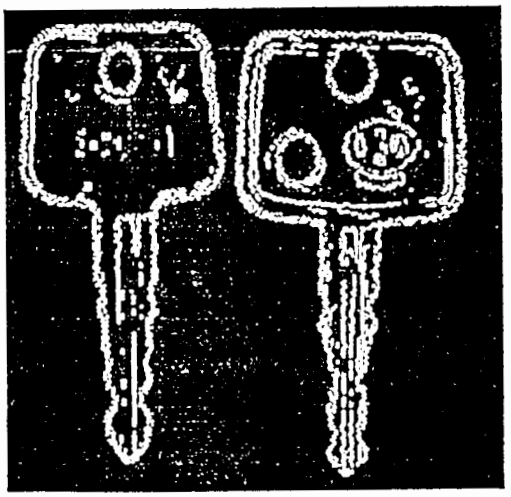
Figure 5.3(a) shows two keys without occlusion, one of the keys having the same boundary shape as the model key. Figure 5.3(b) shows the edge map obtained after applying the two gradient edge operators and the edge thining process on Figure 5.3(a). Figure 5.3(c) is the output of the frame which shows the whole boundary of the key which was recovered. Figure 5.4(a) shows the V-pattern for the key model (the reference axis is the vertical axis). Figure 5.4(b) shows the column on the reference point in the Hough space. For display purposes, the vertical axes in Figure 5.4 have been scaled by the square root of votes (the undesirable problem is that it will exaggerate the errors of low value entries). It is not hard to see that all the basic features were preserved in the accumulated column in the Hough space. Figure 5.5 shows the votes accumulated in the columns on the four immediate neighboring points to the reference point. Apparently, the votes collected in those columns are not significant.

5.2.2 The Occluding Case

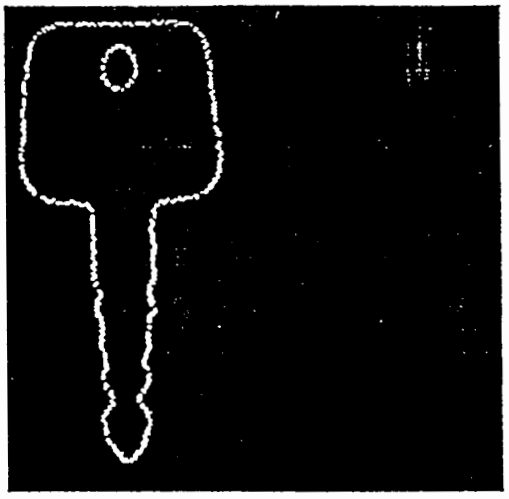
Figure 5.6(a) shows that the key with the same shape as the key model shown in Figure 5.2(a) is partially occluded by another differently shaped key. Our goal is to recover all the unoccluded boundary points of the key with the same shape as the key model.



(a)

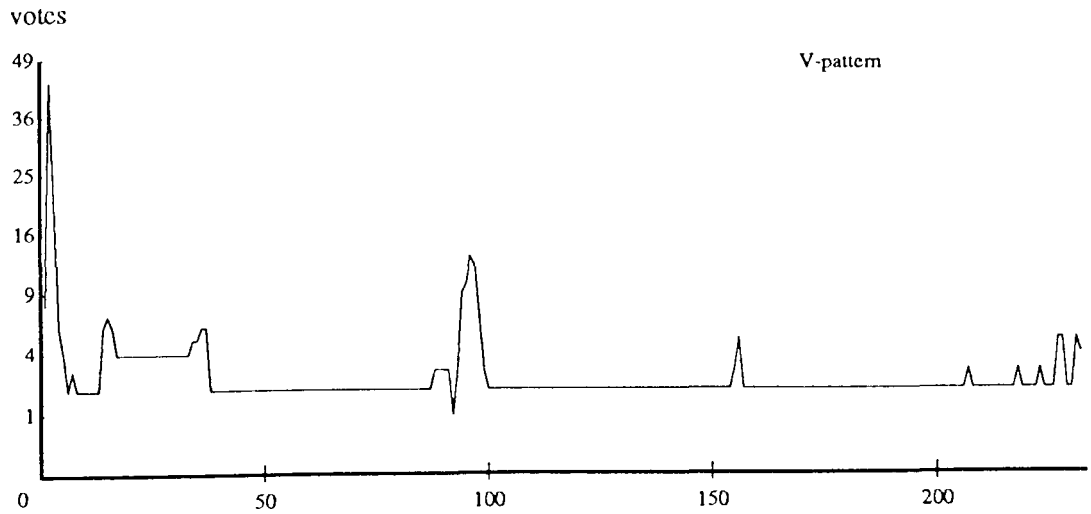


(b)

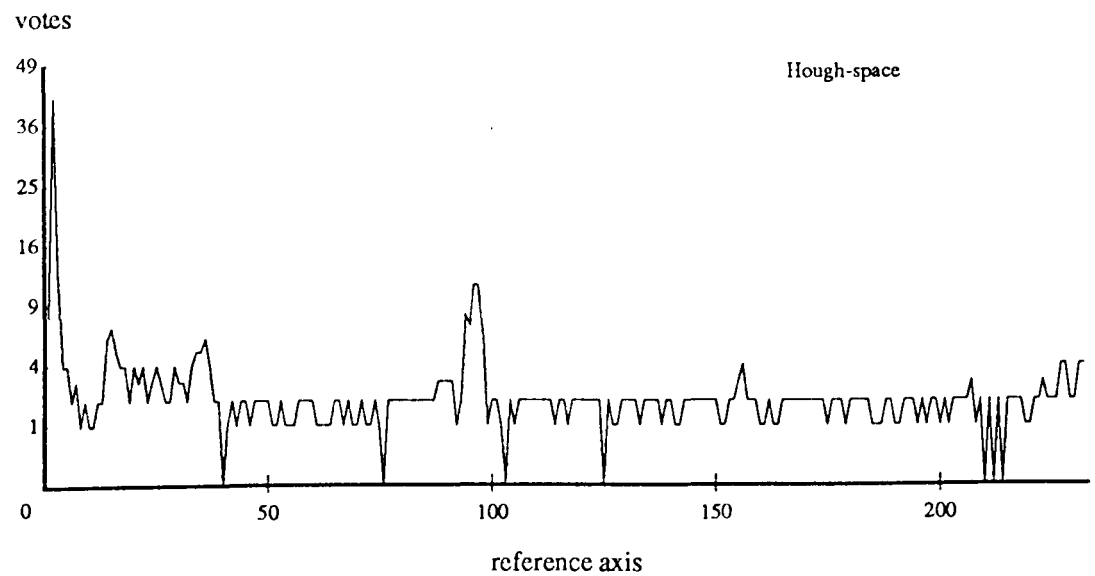


(c)

Figure 5.3 No Occlusion Case



(a) V-pattern Length: 232



(b) The Column on the Reference Point in the Hough Space (Matched: 1-232)

Figure 5.4 V-pattern and the Matching Column in the Hough Space

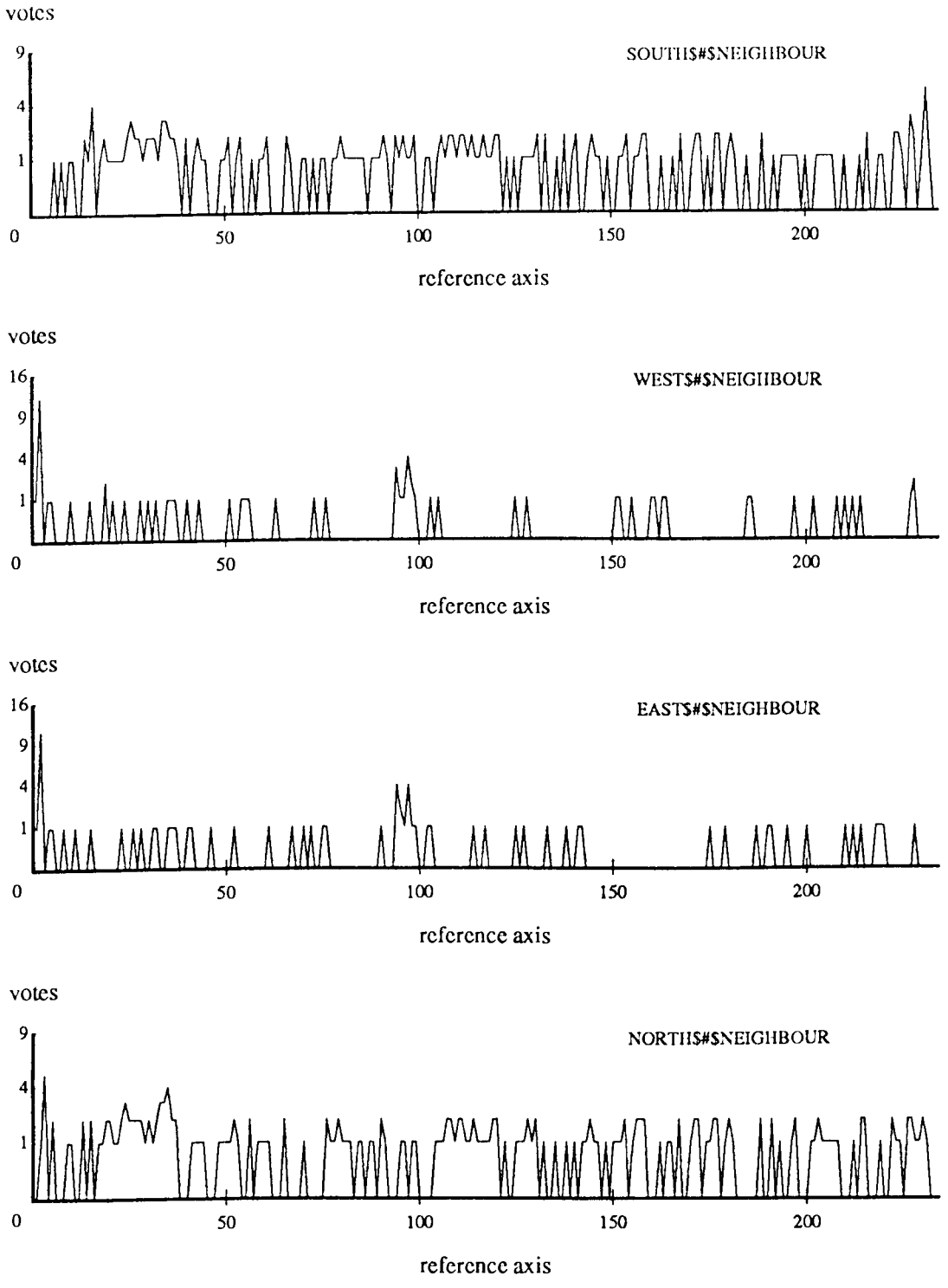
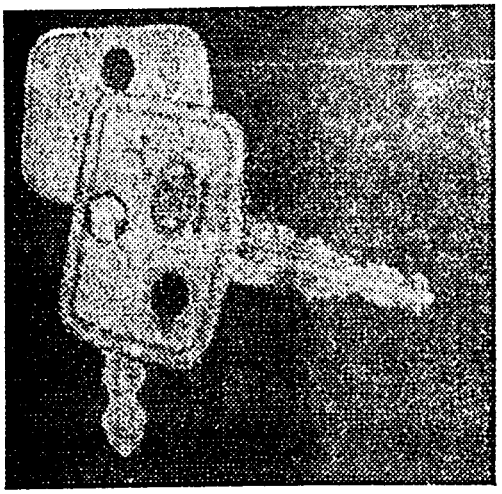


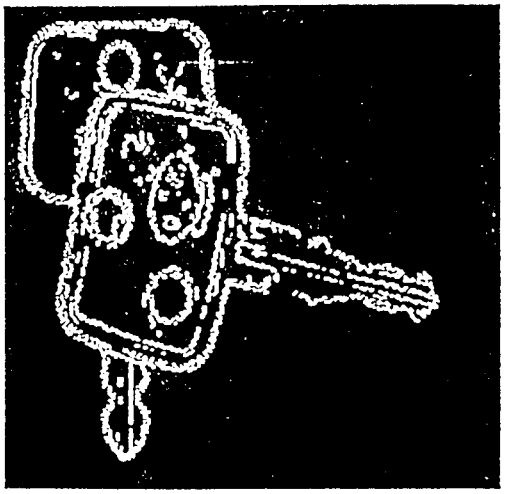
Figure 5.5 Voting Result in Columns of 4 Immediate Neighbors

Ballard's GHT [Ball81] cannot solve such an occlusion problem. The approach taken by Davis [Davi82] is to decompose the model into smaller subtemplates and apply the GHT on these submodels. However, Davis's approach does not suggest how a good decomposition of the model can be achieved so that the GHT could be used effectively to retrieve all the unoccluded parts. His method will leave a difficult model decomposition problem to the users who wanted to use his method to solve a problem like that shown in Figure 5.6(a).

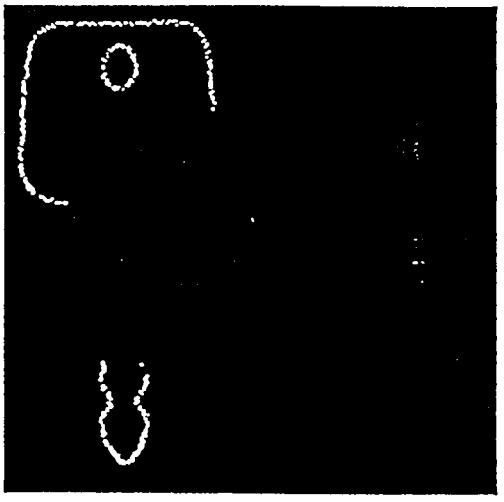
Our system, which is built on the linear Generalized Hough transform technique developed in Chapter 3, has cleanly solved the problem shown in Figure 5.6(a). Figure 5.6(b) is the thinned edge map of Figure 5.6(a). Figure 5.6(c) is the final result in the output frame which contains all the recovered boundary pixels for the key of the same shape as the key model; apparently, it is successful. Figures 5.7(a) and (c) show the V-patterns obtained for the reference axes with 20^0 and 140^0 rotations respectively. Figures 5.7(b) and (d) are the columns on the reference point in the Hough space for the corresponding rotation angles. Figure 5.7(b) corresponds to Figure 5.7(a), and Figure 5.7(d) corresponds to Figure 5.7(c). (Intuitively, the V-pattern for the 20^0 rotation is able to recover the whole bottom piece and partial top piece in Figure 5.6(c), and the V-pattern for the 140^0 rotation is able to discover the whole top piece and partial bottom piece in Figure 5.6(c). The two pieces found separately are combined by an "OR" operation on the recording binary frame.)



(a)

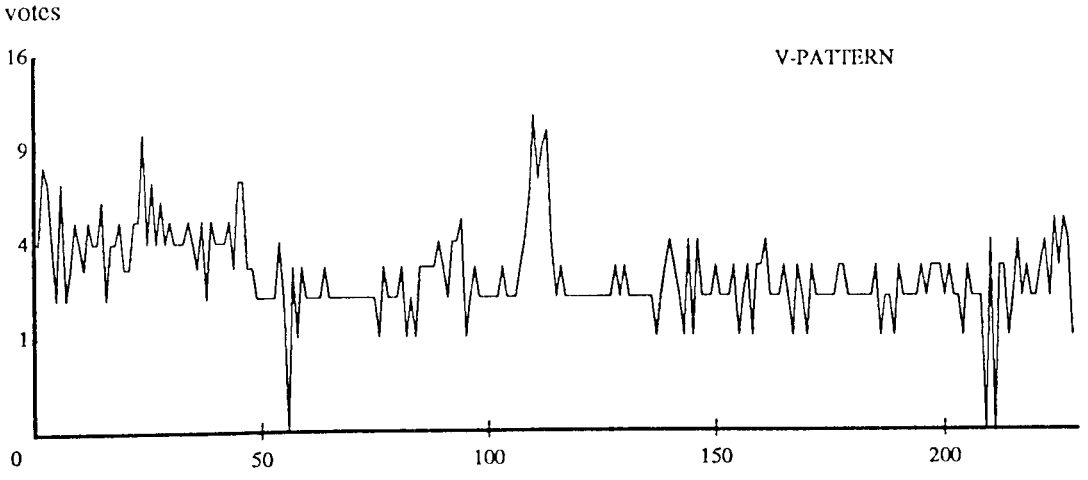


(b)

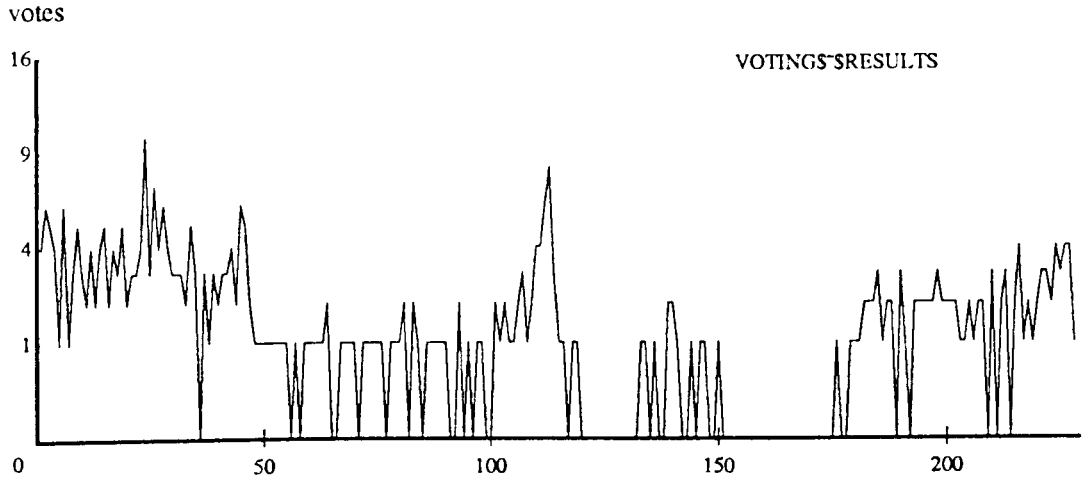


(c)

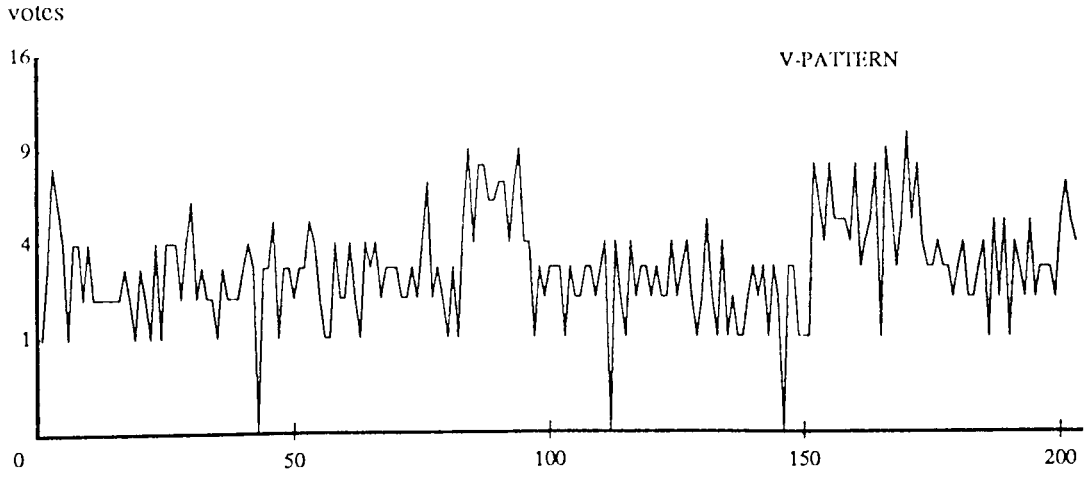
Figure 5.6 Occlusion Case



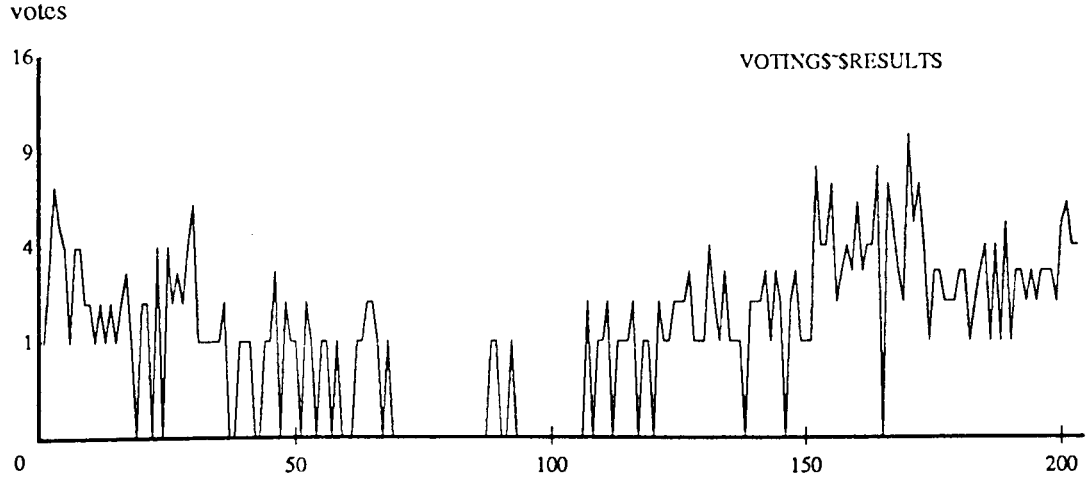
(a) Rotation: 20 degree. Pattern Length: 228



(b) Matched: 1-50, 178-228



(c) Rotation: 140 degree. Pattern Length: 203



(d) Matched: 1-40, 125-203

Figure 5.7 Rotated V-patterns Used for Partial Matches

CHAPTER 6

CONCLUSIONS AND FUTURE RESEARCH

In this thesis, we have shown a useful extension to Ballard's GHT [Ball81]. The essence of the new technique is to use a linear pattern to replace the single reference point used in the GHT.

6.1 Conclusions

The GHT proposed by Ballard has proven to be a very powerful tool in image analysis. It transforms the image points to the parameter combinations in the parameter space by letting the image points vote to the parameter combinations that could have produced them. It then determines the parameter combinations by searching for the peaks in the parameter space. It achieves computational efficiency in detecting arbitrary shapes by such problem transformation. It is not only more computationally efficient than the template matching in the image domain [Davi87a] but also more tolerant to certain shape deformations.

However, the GHT has some inherited problems. The first one is its robustness. It may erroneously declare matches to the shapes similar to the model because the only standard to use for whether a match exists is the peak value. The second problem is in dealing with partially-occluded objects. Davis had suggested shape decomposition to solve the occlusion problem [Davi82]. But his method required "good" decomposition first. The method will still fail if the partial object matches none of the decomposed parts completely.

His approach leaves the difficult task of model decomposition to the users. For example, for the testing images shown in Figure 5.2(a) and Figure 5.6(a), apparently, there would be no good decomposition solutions. The third problem is that the GHT is not suitable for parallel processing. All the points belonging to the shape will vote to the reference point. The reference point becomes a hot spot and causes much contention.

Our new technique has the potential for solving all three problems with the GHT. The linear V-pattern (potentially) increased the robustness of the GHT by requiring that the votes accumulated on the reference point obey a certain linear distribution to declare a match. It also provided a systematic approach to deal with cases of occlusion. Partial matches of V-patterns to the Hough space shows the existence of partial objects. All partial objects can be detected via rotating reference axes. Our experimental system showed that the testing image of Figure 5.6(a) is cleanly solved by our linear generalized Hough transform technique; it would not be solvable by the GHT and would be extremely difficult to solve by Davis's method. We showed that the linear Hough technique achieves this through a reasonable increase over the GHT in space and time complexity.

We have shown how the linear Hough technique can be implemented on two types of Linear Array architecture for parallel processing. The first algorithm, AP-GHT, is based on a very basic linear array processor. We have shown that the speed-up is N/d with N PE's, where d is a pre-determined constant representing the number of rows in the table constructed for the model. We also presented the second algorithm, IAAP-GHT, running on a linear array architecture with independent addressing functions. Our time complexity

analysis for the IAAP-GHT algorithm showed that a linear speed-up is achieved. In other words, when N PE's are used the time complexity is reduced by a factor of N.

6.2 Proposed Future Research

As we have mentioned before, the search for V-patterns in the Hough space should tolerate certain errors by using "non-exact" match scheme for real world images. The practical application of the linear Hough technique in various environments requires the design of heuristic criteria to determine the matches between the V-pattern and the voting patterns in Hough space. We hope that criteria can be found for various applications through statistical studies of images under those conditions, such that the error rate for a practical system using the linear GHT technique is kept at a reasonable level.

The implementation of the linear Hough technique on a more powerful parallel machine to exploit its massive parallel processing power is also another very interesting topic. As we have suggested in Chapter 4, a powerful shared memory architecture will enable parallel processing within each image row as well as image columns. The parallel processing of image columns with a linear array architecture has been discussed in Chapter 4. In Chapter 4, we also showed how concurrent writes to a single shared memory cell could be avoided. We also suggested that a linear speed-up is possible to achieve with an ideal shared memory parallel machine. It is worth studying its potential implementation on the existing parallel machine like the BBN Butterfly where concurrent shared memory accesses are constrained by limited switches. The discussion in section 4.2 shows some promise for solving this problem.

REFERENCES

[Adiv83] Adiv, G. Recovering motion parameters in scenes containing multiple moving objects. *Proceedings of IEEE Conference On Computer Vision and Pattern Recognition*, Washington, pp. 399-400, 1983.

[Aism] AIS-5000 Users' manual, by Applied Intelligent Systems Inc., Ann Arbor, Michigan, USA. 1988.

[Ball81] Ballard, D.H. Generalized the Hough transform to detect arbitrary shapes. *Pattern Recognition*, vol. 13, pp. 111-122, 1981.

[Ball83] Ballard, D.H. and Sabbah, D. Viewer independent shape recognition. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, vol. 5, pp. 653-660, 1983.

[Batc80] Batcher, K.E. Design of a massively praallel processor. *IEEE Transactions on Computers*, vol. 29, No. 9, pp. 826-840, 1980.

[Brow83] Brown, C.M. Inherent bias and noise in the Hough transform. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, vol. 5, pp. 493-505, 1983.

[Cant87] Cantoni, V. and Levialdi, S. Pyramidal systems for computer vision. *NATO AIS Series, Computer and System Science*, vol. 25, Springer Verlag, 1987.

[Cyph97] Cypher, R.E., Sanz, J.L.C. and Snyder, L. The Hough Transform has O(N) complexity on SIMD NxN mesh array architectures. *IEEE Workshop On Proceedings Computer Architectures For PAMI*, Seattle, WA, pp. 115-121, October 1987.

[Dani83] Danielson, P.E. and Ericsson, T.S. LIPP--Proposals for the design of an image processor. *Computing Structures For Image Processing (M.J.B. Duff ed.)*, Academic Press, London, 1983.

[Davi86a] Davies, E.R. Image space transforms for detecting straight edges in industrial images. *Pattern Recognition Letter*, vol. 4, pp. 185-192, 1986.

[Davi86b] Davies, E.R. Corner detection using the generalized Hough transform. *IEE 2nd International Conference on Image Processing and its Applications*, London, pp. 175-179, 1986.

[Davi86c] Davies, E.R. Reduced parameter spaces for polygon detection using the generalized Hough transform. *Proceedings of The 8th International Joint Conference on Pattern Recognition*, Paris, pp. 495-497, 1986.

[Davi87a] Davies, E.R. The performance of the generalized Hough transform: Concavities, ambiguities and positional accuracy. *Proceedings of the 3rd Alvey Vision Conference*, Cambridge, England, pp. 327-334, 1987.

[Davi87b] Davies, E.R. A new framework for analyzing the properties of the generalized Hough transform. *Pattern Recognition Letter*, vol. 6, pp. 1-8, 1987.

[Davi87c] Davies, E.R. Improved localization in a generalized Hough scheme for the detection of straight edges, *Image and Vision Computing*, vol. 5, No. 4, pp. 279-286, 1987.

[Davi82] Davis, L.S. Hierarchical generalized Hough transforms and line segment based generalized Hough transforms. *Pattern Recognition*, vol. 15, pp. 277-285, 1982.

[Davi84] Davis, R. and Thomas, D. Systolic array chip matches the pace of high-speed processing. *Electronic Design*, pp. 207-218, October, 1984.

[Dhom86] Dhome, M. and Kasvand, T. Hierarchical approach for polyhedra recognition by hypothesis accumulation. *Proceedings of The 8th International Joint Conference on Pattern Recognition*, Paris, pp. 88-91, 1986.

[Duda72] Duda, R.D. and Hart, P.E. Use of the Hough transform to detect lines and curves in pictures. *Communications of The ACM*, vol. 15, pp. 11-15, 1972.

[Duda73] Duda, R.D. and Hart, P.E. Pattern Classification and scene analysis. Wiley, New York, 1973.

[Duff73] Duff, M.J.B., Watson, D.M., Fountain, T.J., and Shaw, G.K. A cellular logic array for image processing. *Pattern Recognition*, vol. 5, pp. 229-247, 1973.

[Duff78] Duff, M.J.B. Review of the CLIP Image Processing System. *National Computer Conference*, Anaheim, CA, 1978.

[Duff85] Duff, M.J.B. Real applications on CLIP4. *Integrated Technology for Parallel Image Processing*, S. Levialdi (ed.), Academic Press, 1985.

[Duff86] Duff, M.J.B. ed. Intermediate-level image processing, *Academic Press*, London, 1986.

[Dyer81] Dyer, C.R. and Rosenfeld, A. Parallel image processing by memory-augmented cellular automata, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 3, pp. 29-41, 1981.

[Fish85] Fisher, A.L. and Highman, P.T. Real time image procession on scan line array processors. *IEEE Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, Miama, FL, pp. 484-489, 1985.

[Foun86] Fountain, T.J. Array architectures for iconic and symbolic image processing. *Proceedings of The 8th International Joint Conference on Pattern Recognition*, Paris, pp. 24-33, 1986.

[Grim88] Grimson, W.E.L. and Huttenlocher, D.P. On the sensitivity of the Hough transform for object recognition. *Proceedings of the 2nd International Conference on Computer Vision*, pp. 700-706, 1988.

[Grow85] Growther, W., Goodhue, J., Starr, E., Thomas, R., Miliken, W., and Blackadar, T. Performance measurements on a 128-node butterfly parallel processor. *Proceedings of the 1985 IEEE International Conference on Parallel Processing*, Pennsylvania, PA, pp. 531-540, 1985.

[Haka84] Hakalahti, H., Harwood, D., and Davis, L.S. Two-dimensional object recognition by matching local properties of contour points, *Pattern Recognition Letter*, vol. 2, pp. 227-234, 1984.

[Hend84] Henderson, T.C. and Fai, W.S. The 3D Hough shape transform, *Pattern Recognition Letter*, vol. 2, pp. 235-238, 1984.

[Hill85] Hills, W.D. The connection machine. *The MIT Press, Cambridge, MA*, 1985.

[Houg62] Hough, P.V.C. A method and means for recognizing complex patters, *USA Patent 3,069,654*. 1962.

[Ibra86] Ibrahim, H.A.H., Kender, J.R., and Shaw, D.E. On the application of massively parallel SIMD tree machines to certain intermediate-level vision tasks. *Computer Vision, Graphics, and Image Processing*, vol. 36, pp. 53-75, 1986.

[Illi88] Illingworth, J. and Kittler, J. A survey of the Hough transform. *Computer Vision, Graphics, and Image Processing*, vol. 44, pp. 87-116, 1988.

[Kasi83] Kasif, S., Kitchen, L., and Rosenfeld, A. A Hough transform technique for subgraph isomorphism, *Pattern Recognition Letter*, vol. 2, pp. 83-88, 1983.

[Kimm75] Kimme, C., Ballard, D.H., and Sklansky, J. Finding circles by an array of accumulators. *Communications of The ACM*, vol. 18, pp. 120-122, 1975.

[Levi72] Levialdi, S. On shrinking binary picture patterns. *Communications of the ACM*, vol. 15, No. 1, pp. 7-10, 1972.

[Li90] Li, Z.N., Tong, F., and Laughlin, R.G. Parallel algorithms for line detection on a 1xN array processor. *Technical Report*, Center of System Science, Simon Fraser University, 1990.

[Litt87] Little, J.J., Blelloch, G., and Cass, T. Parallel algorithms for computer vision on the connection machine. *Proceedings of the 1st IEEE International Conference on Computer Vision*, London, pp. 587-591, 1987.

[Mark80] Marks, P. Low level vision using an array processor. *Computer Graphics Image Processing*, vol. 14, pp. 281-292, 1980.

[Matt86] Matthews, K.N. The CLIP7 image analyser - a multi-bit processor array. *PH.D Dissertation*, University of London, 1986.

[Merl75] Merlin, P.M. and Farber, D.J. A parallel mechanism for detecting curves in pictures, *IEEE Transactions On Computers*, vol. 24, pp. 96-98, 1975.

[Mill85] Miller, R. and Stout, Q. Geometric Algorithms for digitized pictures on a mesh-connected computer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 7, No. 3, pp. 216-228, 1985.

[Mori85] Moring, I. and Hakalahti, H. Scale independent method for object recognition. *Proceedings of Scandinavian Conference On Image Processing*, Trondheim, pp. 103-114, 1985.

[Nage72] Nagel, R.N. and Rosenfeld, A. Ordered search techniques in template matching. *Proc. IEEE* vol.60, pp. 242-244, 1972.

[Nass80] Nassimi, D. and Sahni, S. Finding connected components and connected ones on a mesh-connected parallel computer. *SIAM Journal of Computing*, vol. 9, No. 4, pp. 744-757, 1980.

[Olso87] Olson, T.J., Bukys, L., and Brown, C.M. Low level image analysis on an MIMD architecture. *Proceedings of the 1st IEEE International Conference on Computer Vision*, London, pp. 468-475, 1987.

[Perk78] Perkins, W.A. A model based vision system for industrial parts. *IEEE Transactions on Computers*, vol. 27, pp.126-143, 1978.

[Perk80] Perkins, W.A. Simplified model-based part locator. *Proceedings of the 5th International Conference on Pattern Recognition*, December, 1980.

[Pott83] Potter, T.J. Image processing on the massively parallel processor. *IEEE Computer*, pp. 62-67, 1983.

[Pott85] Potter, J.L. (ED) *The massively parallel processor.* MIT Press, Cambridge, MA, 1985.

[Reev84] Reeves, A.P. Survey: parallel computer architectures for image processing. *Computer Vision, Graphics, and Image Processing*, vol. 25, pp. 68-88, 1984.

[Rose69] Rosenfeld, A. *Picture Processing by Computer*, Academic Press, New York/Lodon, 1969.

[Rose83] Rosenfeld, A. Parallel image processing using cellular arrays. *IEEE Computer*, pp. 14-20, 1983.

[Rose88] Rosenfeld, A., Ornelas, J. (Jr.), and Hung, Y. Hough transform algorithm for mesh-connected SIMD parallel processors. *Computer Vision, Graphics, and Image Processing*, vol. 41, pp. 293-305, 1988.

[Schm88] Schmitt, L.A. and Wilson, S.S. The AIS-5000 Parallel Processor. IEEE Trans. On Pattern Analysis And Machine Intelligence, vol. 10, No. 3, pp. 320-329, May 1988.

[Silb84] Silberberg, T.M., Davis, L., and Harwood, D. An iterative Hough procedure for 3D object recognition. *Pattern Recognition*, vol. 17, pp. 621-629, 1984.

[Ston71] Stone, H. Parallel processing with a perfect shuffle. *IEEE Transactions on Computers*, vol. 20, pp. 153, 1971.

[Tani87] Tanimoto, S., Ligocki, T., and Ling, R. A prototype Pyramid machine for hierarchical cellular logic. *Uhr, L. (ed.), Parallel Hierarchical Computer Vision*, Academic Press, Orlando, FL, 1987.

[Tsuj78] Tsuji, S. and Matsumoto, F. Detection of ellipses by modified Hough transformation. *IEEE Transactions On Computers*, vol. 27, pp. 777-781, 1978.

[Tsuk83] Tsukune, H. and Goto, K. Extracting elliptical figures from an edge vector field, *Proceedings of IEEE Conference On Computer Vision and Pattern Recognition*, Washington, pp. 138-141, 1983.

[Turn85] Turney, J.L., Mudge, T.N., and Volz, R.A. Recognizing partially occluded parts. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, vol. 7, pp. 410-421, 1985.

[Unge58] Unger, S.H. A computer oriented toward spatial problems. *Proc. IRE*, vol. 46, pp. 1744-1750, 1958.

[Vand77] VanderBrug, G.J. and Rosenfeld, A. Two-stage template matching. *IEEE Transactions on Computers*, vol. 26, No. 4, pp. 384-393, 1977.

[Wall85] Wallace, R.S. A modified Hough transform for lines. *Proceedings of IEEE Conference On Computer Vision and Pattern Recognition*, San Francisco, pp. 665-667, 1985.

[Will79] Willie, J.C. The complexity of Parallel computation. *Ph.D Dissertation*, Cornell University, Ithaca, NY, August, 1979.

[Wils88] Wilson, S.S. One Dimensional SIMD Architectures--The AIS-5000. *Multicomputer Vision*, Academic Press, London, pp. 131-149, 1988.