NOTICE

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

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

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

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

AVIS

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

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

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

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

A New Perspective of Hough Transform: Theory and Applications

Derek Chi Wai Pao

A Thesis

in

The Department

of

Computer Science

Presented in Partial Fulfilment of the Requirements
for the Degree of Doctor of Philosophy at
Concordia University
Montreal, Quebec, Canada

June 1991

ISBN 0-315-68779-7

ABSTRACT

A New Perspective of Hough Transform: Theory and Applications

Derek Chi Wai Pao, Ph.D.
Concordia University, 1991.

The Hough transform is an effective method to detect parametric curves in binary images. However, some of its drawbacks are also well known. For the detection of straight lines, the method only determines the parametric representation; information concerning the exact locations and lengths of the line segments is not available. The lack of the positional information may limit the usefulness of the extracted features in some applications. Also, because of its statistical nature, spurious lines may be detected. For the detection of higher order parametric curves, the computational complexity and memory space requirement of the method grow exponentially with the number of parameters of the curves. In this thesis, we present novel methods to detect straight lines and higher order parametric curves, such as circles and ellipses.

For the detection of straight lines, we present a modified algorithm (SLHT) that incorporates a pixel connectivity check. By so doing, the end points of the line segments can be computed, and unrelated pixels (noise or pixels of far apart objects) can be filtered out. A new line selection algorithm based on comparisons among neighboring accumulator cells is developed to replace the simple thresholding method. Short lines as well as long lines can be detected, but spurious lines are totally eliminated. The modified Hough transform is implemented on a systolic array. The area-time complexity of the proposed architecture compares favorably to other parallel designs.

For the detection of higher order parametric curves, we present novel methods based on the decomposition of the high dimensional parameter space. In our approach,

the SLHT is first computed. We then compute other transforms from the θ–ρ space (θ and ρ are the normal parameters of a straight line) to the subspaces of the curve. Conceptually, each point in θ–ρ corresponds to a straight line in the image plane; and the line is tangent to the curve. The major advantage of formulating the transform using the set of tangents is that it allows the translation parameters be easily separated from the other parameters of the curve. Also, rotation in the image plane corresponds to circular shifting in the θ–ρ space. Novel methods to detect circles and ellipses are presented. The computational complexities of our methods compare favourably to other Hough based algorithms. The approach is further generalized to recognize arbitrary shapes by computing a signature in the θ–ρ space. This signature is invariant to translation and easily normalized with respect to the size of the object. Rotation in the image plane corresponds to circular shifting of the signature.

In our curves detection methodology, we assume that the shape is being represented by its complete set of tangents. In this thesis, we establish the theoretical foundation of our methodology by proving the uniqueness of the tangent set representation of closed smooth curves. A byproduct of our proof is that it is possible to fully reconstruct a curve from its complete set of tangents without knowing the points of tangency.

The curves detection techniques are applied to extract graphic features for automatic conversion of engineering line drawings. The graphic features extraction problem is formulated as a segmentation problem. A stepwise segmentation refinement strategy is developed in which the Hough transform is applied to generate initial hypotheses to guide the segmentation process. Good segmentation results were obtained in the preliminary experimentation.

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere gratitude to my supervisors, Prof. Hon F. Li and Dr. R. Jayakumar, for their guidance, support, and encouragement throughout this research. Even though the high standard of requirements of my supervisors is not easy to live with, I will not hesitate to admit that I have benefited enormously from their critical comments and insights. Besides learning how to define and tackle research problems, I was also given vigorous training on how to write research papers and how to make oral presentations. These technical skills are extremely useful and essential to the success of my future career. I am very lucky to have not just one but two excellent supervisors.

From the very beginning of my Ph.D. program, I have been associated with the Optical Character Recognition (OCR) research group of the Centre of Pattern Recognition and Machine Intelligence (CENPARMI) under the leadership of Prof. Ching Y. Suen. I am very grateful to Prof. Suen and other members of the OCR group, which includes Dr. Louisa Lam, Raymond Legualt, and Christine Nadal, for their support and encouragement. I benefited a lot from the stimulating discussions in the regular meetings and their constructive comments.

I would also like to thank Prof. R. M. H. Cheng and Mr. Simon Poon of the Mechanical Engineering Deptartment of Concordia University for helping me to gain a better understanding of engineering line drawings.

I am very thankful to the technical staff of the Computer Science Department, Mr. Paul Gill, Mr. Girish Patel, Mr. William Wong, and Mr. Mike Yu for their technical advice and assistance during my experimentations. In particular, I am very impressed by the enthusiasm of Mr. Paul Gill in helping students to solve system problems.

Throughout the course of my graduate studies at Concordia University, I received much support from my friends and colleagues. Their help in both academic matters and daily life problems are very much appreciated.

Last but not least, I would like to thank the examiners, in particular the external examiner Dr. Steven L. Tanimoto and the external-to-program examiner Dr. R. M. H. Cheng, for their valuable and constructive comments on the thesis.

This research is partly supported by research grants from the Natural Science and Engineering Research Council of Canada (NSERC) and partly by a Strategic Research Grant from NSERC.

TABLE OF CONTENTS

LIST OF FIGURES

Figures

LIST OF TABLES

Tables

Chapter 1
Introduction

Detection of straight lines and higher order parametric curves such as circles and ellipses in binary images of man made objects is a fundamental problem in computer vision. These entities are often used as features in higher level applications such as object recognition, automatic inspection and vehicle guidance, etc. One effective method to detect these features is the Hough transform [Hou62, Dud75]. In this method, the curve to be detected is described by a characteristic equation $f((a_1,a_2,\ldots,a_n),(x,y)) = 0$, where $(a_1,\ldots,a_n)$ are the parameters of the curve and (x,y) are the free variables. By interchanging the roles of the parameters and variables, an image point (x_i,y_i) is mapped into a hypersurface in the n-dimensional parameter space $R^n = A_1 \times A_2 \times \cdots \times A_n$ using the above equation. Hypersurfaces corresponding to image points that belong to the same curve will intersect at a common point in the parameter space. In computing the transform, the parameter space is quantized and represented by an accumulator array. Each accumulator cell keeps track of the number of image points that are mapped onto it. A high vote count represents high level of evidence of the presence of the curve instance in the image, and the indices of the accumulator cell reveal the parameters of the curve instance. Hence, the problem of detecting curves in the image space is equivalent to a much simpler problem of locating peaks in the quantized parameter space.

The Hough transform has several desirable properties. First, it operates on raw binary images with minimal preprocessing requirements. Second, each image point is treated independently, and therefore the method can be implemented in parallel

machines to speed up the computation. Third, its independent accumulation of evidence means that it can recognize partial or occluded shapes. Also, random noise in the image is unlikely to contribute coherently to a single bin of the accumulator array, and therefore, the method is, to a certain degree, insensitive to random noise in the image.

In the past decade, there have been a large number of publications on the Hough transform [Ill88], and it was the major theme of several Ph.D. dissertations [Kri87b, Ku87, RiJS89]. Applications of the Hough transform include polyhedral objects recognition [Eng88], vehicle guidance [Ini84], target detection and tracking [Cow83, Kri87a], automatic inspection [Dye83, Hua85, Shu87], image segmentation [Jay83, Poe86], image registration [Anu85, Yam81], character recognition [Kus85, Che89], subgraph isomorphism [KaS83], motion estimation [Rad86], recognition of 3D objects using range data [Kri89, Tay90], satellite image processing [Lee89], and document processing [Fle88, Hin90]. In this thesis, we focus on the detection of 2D planar curves using the Hough transform.

The Hough transform is basically a statistical summary of the likelihood of presence of the features in the image. If the features being looked for are parametric curves, then the method only determines the possible parameters of the curve instances in the image, for example the slope and intercept or the normal parameters, ρ and θ, of the straight lines. The lack of information regarding the exact location and length of the line segments may limit the usefulness of the extracted features. A more severe problem, known as the sensitivity problem, arises when the number of features in the image is large, and the vote patterns of distinct features interfere with each other and lead to the formation of spurious peaks.

Another drawback of the Hough transform is its high computational cost and memory (space) requirement. The computational cost can be divided into two components, the cost of accumulating the votes and the cost of detecting peaks in the

parameter space. If the curve to be detected is characterized by n parameters and if each parameter is quantized into M distinct levels, then the voting cost is $O(PM^{n-1})$, where P is the number of pixels in the image. The computational cost of detecting peaks in the n-dimensional parameter space is normally $O(M^n)$ since every accumulator cell has to be inspected at least once. The amount of storage required is also normally $O(M^n)$. If either M or n is large, then the computation cost and memory requirement may become prohibitive. For example, detection of ellipses, which are being characterized by five parameters, the center coordinates, the major and minor axes, and rotation, is often deemed infeasible using this method.

In this thesis, we develop a modified Hough transform for straight lines detection that will incorporate a pixel connectivity check. By so doing, confusions that arise from the statistical nature of Hough transform can be avoided. Spurious peaks can be totally eliminated. Each detected line segment is characterized by its two endpoints in addition to the normal parameters ρ and θ. The modified Hough transform is implemented using a systolic array. The merits of our modifications and the systolic implementation are listed below.

1. Pixel connectivity can be checked incrementally such that only a constant amount of memory and a simple comparator are required in each processing element (PE).
2. No multiplication is required in the computation. Hence, the design of the PEs is much simplified.
3. The area-time (AT) complexity of our design compares favourably with the conventional sequential algorithm and other parallel algorithms.

For the detection of higher order curves, such as circles and ellipses, we develop a novel parameter space decomposition approach that will reduce the time and space complexity substantially. Our method is based on the shape representation scheme that uses the set of lines tangent to the curve. The use of this representation scheme for

object recognition was first studied in [Cas87] and later in [McK90]. The major advantage of this representation scheme is that it allows us to separate the translation and rotation parameters from the intrinsic parameters of the curve. Let L be the set of lines tangent to the curve C. The pattern obtained by plotting the normal parameters of each line $l \in L$ in the θ–ρ plane is referred tc as the "transform function" of the curve C, denoted as $\rho(\theta)$. In the discrete domain, this transform function can be computed using the modified Hough transform we have developed. Each detected line segment is an approximate tangent to the curve.

Let $\rho_0(\theta)$ be the transform function of a curve C located at the origin of the x–y plane. If C' is an instance of C which is rotated by an angle ϕ in the counterclockwise direction and translated to (x_0,y_0), then the transform function of C' is given by

$$\rho(\theta) = t\cos(\theta-\alpha) + sign * \rho_0(\theta-\phi)$$

where $t = \sqrt{x_0^2+y_0^2}$, $\alpha = \tan^{-1}(y_0/x_0)$, and $sign = 1$ if $\theta \geq \phi$; otherwise, $sign = -1$. The $t\cos(\theta-\alpha)$ term is called the *translation term* and the $sign * \rho_0(\theta-\phi)$ term is called the *intrinsic term*. Mathematical derivation of the transform function will be given in chapter 3. One desirable property of this representation is that it provides a natural way to decompose the high dimensional parameter space. By decoupling the two terms, we can decompose the high dimensional parameter space into three subspaces, namely the translation, rotation and intrinsic parameters.

Let's consider the detection of ellipses. In one of our methods (which will be elaborated in chapter 4), we assume the orientation of the ellipse and eliminate the intrinsic term. A voting phase in the t–α plane is carried out to determine the center coordinates of the ellipses. Knowing the orientation and center coordinates, we can perform another voting phase in the a–b plane (a and b are the major and minor axes of the ellipse, respectively) to determine the major and minor axes. If the orientation is not known, then a search on the ϕ axis is carried out. Our method only requires searching

a 1D space and voting on 2D spaces. Substantial savings in both time and memory are achieved when compared with the conventional Hough transform that involves voting on a 5D space.

In our methods and that of Casasent and Krishnapuram, the curve is characterized by its set of tangents. One fundamental question we ask is whether this representation of a curve is unique or not. In other words, can we have two distinct curves that can have exactly the same set of tangent lines? In this thesis, we present a mathematical proof based on convexity theory that a 2D continuous, smooth, simple closed curve is indeed uniquely characterized by its set of tangents. A byproduct of our proof is that reconstruction of a closed smooth curve from its set of tangents is feasible.

Observing from the transform function $\rho(\theta)$, if we can eliminate the translation term, then we are able to obtain a signature of the curve which will depend only on the rotation and intrinsic parameters. This signature has three very useful properties: (i) it is invariant to translation of the curve in the image plane; (ii) rotation of the curve corresponds to circular shifting of the signature; and (iii) the scaling factor of the signature can be easily normalized. This signature is, however, sensitive to variations of image patterns. Recognition of the curve only involves computing a 1D convolution of the signature of the curve with that of the reference curve. We call this signature the *Scalable Translation Invariant Rotation-to-Shifting* (STIRS) signature. The computational complexity of this matching method is much lower than the conventional template matching method. This method is especially attractive when the test pattern has to be matched against a large number of reference patterns.

To demonstrate the applicability of our curve detection techniques, we apply our method to extract graphic features from engineering line drawings. Automatic conversion of engineering line drawings is essentially a computer vision problem. The ultimate goal is to derive an intelligent interpretation of the drawing that can support efficient editing, 3D modeling and integration with CAM (computer aided

manufacturing). Extracting meaningful geometrical features is very different from computing a "best" approximation to the set of data points. For example, a curve can be approximated by a set of straight line segments. But this representation is by no means adequate for the above stated objectives. The basic question is not how to obtain a set of curves that can accurately approximate the input image, but rather how to infer a meaningful description of the input image in terms of the set of extracted features. Our view on graphic feature extraction is essentially a segmentation problem. Once we can partition the set of data points into a number of subsets, then it won't be difficult to derive the best parametric representation of a curve that passes through a particular subset of data points. A stepwise segmentation refinement strategy based on the Hough transform is developed. In this approach, the dominant features in the image are successively extracted, and the segmentation is refined after each iteration. Very good segmentation results have been achieved in the preliminary experimentation. A more detailed introduction to the engineering line drawing conversion problem is given in chapter 6.

The thesis is organized as follows. In chapter 2, we present our modified Hough transform for the detection of straight lines and its systolic implementation. In chapter 3, we establish the theoretical foundation of our curves detection methodology by proving the uniqueness of the shape representation scheme using the set of tangents. The parameter space decomposition approach to detect circles and ellipses is discussed in chapter 4. Formal definition and application of the STIRS signature to the recognition of arbitrary shapes are presented in chapter 5. Application of the curve detection techniques to the engineering line drawing conversion problem is presented in chapter 6. Finally, some concluding remarks and discussion on future research are given in chapter 7.

Chapter 2
Detection of Straight Lines

A straight line l can be represented by the equation

$$l: \ \rho = x\cos\theta + y\sin\theta$$

where θ is the angle between the normal of the line and the x-axis, and ρ is the perpendicular distance of the origin from the line l (figure 2.1). Let the size of the input image be $N \times N$. Then the value of θ is restricted to $[0°,180°)$, and ρ lies within the range $[-\sqrt{2}N, \sqrt{2}N]$. A straight line l is uniquely characterized by the tuple (θ,ρ). The θ–ρ space is quantized and represented by a 2D accumulator array of size $m_\theta \times m_\rho$. The quantizations in ρ and θ are $\Delta\rho$ and $\Delta\theta$ units, respectively. To detect straight lines in the image, every image point (x_i, y_i) is mapped into a sinusoidal curve in the θ–ρ space by

$$\rho = x_i\cos\theta + y_i\sin\theta.$$

The accumulator cells (buckets) along the sinusoids are incremented by an amount depending on the scheme, often a function of the pixel value. In this thesis, we only consider binary images, and the accumulator cell is incremented by 1. The other way to look at the voting process is that each bucket corresponds to a rectangular window in the image plane with orientation $\theta + 90°$ and width $\Delta\rho$. The vote count of the bucket is equal to the number of image pixels lying within the corresponding window. Peaks are located by thresholding the accumulator array and they are regarded as hypotheses of the presence of straight lines in the image.

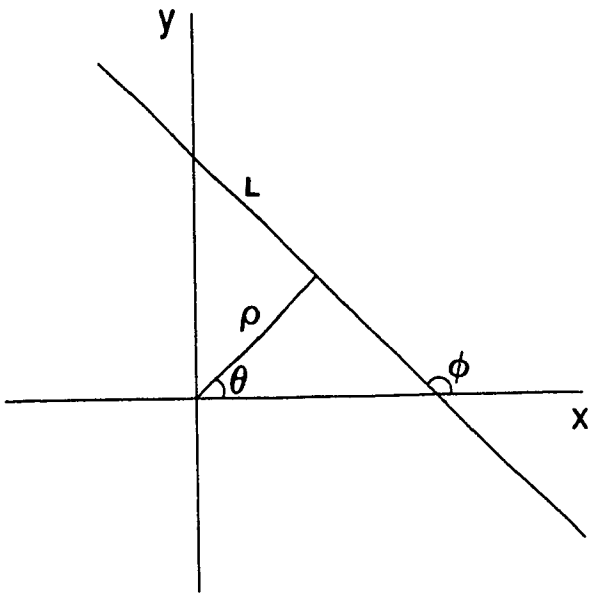


Figure 2.1 Normal parameters of a straight line.

Because of quantizations in the image space and parameter space, the sinusoid of the image pixels belonging to a line instance do not, in general, intersect precisely at a common point in the parameter space. Moreover, each accumulator cell will also accumulate votes due to noise and unrelated pixels in the image. Hence, the vote count does not necessarily reflect the true level of evidence. Selection of the threshold value is another difficult problem. If the threshold is set too high, short lines cannot be detected. On the other hand, if the threshold is too low, spurious peaks may result.

Several studies on the behaviour of the Hough transform in relation to the choice of quantization of the parameter space and the effect of noise on parameter accuracy have been reported in the literature [Sha75, 78,79]. Van Veen and Goren studied the quantization errors in straight line detection using the (θ,ρ) parameterization [Van81]. Analytic results for the spreading of peaks as a function of quantization of the image space and θ–ρ space were derived. These results serve as guidelines in choosing the quantization of the parameter space to minimize spreading of peaks. Leavers and

Boyce analyzed the shape of peaks in the θ–ρ space [Lea87]. They observed that peaks in the θ–ρ space due to straight lines exhibit a characteristic butterfly shape, and suggested suitable convolution filters which would enhance these peaks. Niblack and Petkovic also attempted to improve the accuracy of the estimated parameters by filtering the θ–ρ space [Nib88].

Another approach to improve the performance is by reducing the background votes. One method proposed by Brown is known as the complementary Hough, CHough [Bro83]. The Hough transform is recast as a linear imaging problem. The pattern of votes generated in the parameter space by each image point is called the feature point spread function, psf. The final accumulator is the superposition of the feature psf's in the image. In this approach, an image point contributes positively to some parameter values, but negatively to the off-peak parameter values. The net effect is to reduce the mean and variance of the background votes and to produce more prominent peaks. However, this method only applies to a psf that is symmetric, and may suffer more from quantization effects.

Another method to reduce background votes is by incorporating additional constraints in the mapping of each image point to parameter space. One method is by the use of gradient information associated with the image pixels [Bal81]. For example in straight line detection, if the gradient of an image point p is g, then the point p will only vote for the accumulator cells in columns

$$\theta = ((\tan^{-1}(g \pm \varepsilon) + 90°) / \Delta\theta) \bmod m_\theta$$

where ε is the expected error in the measured gradient g. The other method is to consider pairs of pixels. Random combinations of pairs of pixels, (p_1 and p_2), are generated. Each pair of pixels will vote for the parameters that correspond to the line passing through the two points. The effectiveness of this method degrades substantially with increased complexity of the image. Suppose the line l has k pixels and the total

number of pixels in the image is P. Then the chance of getting a pair of points from the line l is $(k/P)^2$. So, when $P \gg k$, the number of votes the line l can have with a fixed number of trials decreases quadratically with the complexity of the image.

The above mentioned attempts in improving the accuracy and sensitivity of the Hough transform for straight lines detection only achieve limited success. The major reason for that is because of the inherent statistical nature of the Hough transform. The votes due to noise and unrelated pixels cannot be distinguished from the votes due to genuine pixels. A vote count of k may be due to a line with k pixels or as well be due to k noise pixels that happens to be collinear. One way to overcome this problem is by incorporating a pixel connectivity check, such that the accumulator cell can distinguish between noise and genuine pixels. A straightforward way to implement pixel connectivity requires each accumulator cell to store the coordinates of the pixels that vote for it. A better method was proposed by Shu et al [Shu87] which used a bit array in each accumulator cell to determine the connectivity and length of the line segments. Every pixel mapped to an accumulator cell will mark the corresponding bit in the bit array. A continuous string of marked bits in the bit array corresponds to a group of connected collinear pixels. This method simplifies the connectivity check but the memory requirement is still substantial because the size of the bit array grows in proportion with the size of the input image.

In this chapter, we introduce a modified Hough transform which performs a connectivity check in a simple and efficient way. A systolic architecture [Kun82, LPJ89b] that implements this modified transform is developed. The systolic array takes the bitmap of the binary image as input and processes one row/column of pixels concurrently. The processing elements (PEs) of the array are very simple and the area-time complexity of the proposed architecture compares favourably to other parallel implementations. By checking the contiguity of edge pixels, unrelated pixels can be filtered away. Each line segment is characterized by its two end points, and a bucket

can hold multiple collinear lines. Knowing the end points of a line, we can determine the "best" fit to the group of "collinear" edge pixels by counter-checking with other lines that are detected in adjacent buckets. This approach to selecting the best lines can avoid the confusion that usually arises when simple thresholding is applied.

2.1 Modified Hough Transform and a Systolic Implementation

In our approach, we check the pixel connectivity incrementally. Suppose the image is being scanned from left to right and bottom to top for $45° \leq \theta \leq 135°$, and scanned from bottom to top and left to right for other values of θ. The edge pixels belonging to the same line segment will be processed and mapped to a common bucket in sorted order. Each bucket can reject unrelated pixels by simply checking the connectivity of the incoming pixel with the last pixel sitting in the bucket. This approach only requires a constant amount of memory space to store the start and end points of the line segment in each accumulator cell and a simple comparator for checking connectivity. Whenever a discontinuity[1] is detected, the current line segment is terminated and the incoming pixel defines the starting point of a new line segment. Isolated (single) pixels and very short line segments whose lengths are less than some threshold value, say L_{min}, can be simply discarded.

When mapping this approach into a systolic array for implementation in VLSI [Mea80], we want to achieve the following three objectives:

1. *Minimize external and intercell I/O requirements:* To reduce external I/O requirements, we process the bit map of the binary image directly. Each pixel is represented by a single bit instead of its coordinates. By using a carefully designed local timing scheme, the accumulator cell can compute the location of the

[1] 8-connectivity is used in our work. However, the connectivity constraints can be relaxed a little to allow for small gaps of predefined size along the line.

pixel based on its arrival time (this will be elaborated later in this section). This means that we don't have to pass around the coordinates of the edge pixels. This represents a huge saving in intercell data routing and consequently in space complexity.

2. *Simple processing cell:* Hardware multipliers are complicated. In our design, all multiplications are replaced by simple additions. Hence the cell design is extremely simple.
3. *Maximize concurrency:* The algorithm exploits concurrency by both multiprocessing and pipelining of the processing cells. One column or row of pixels are processed per time unit (the time required to perform one basic arithmetic-logic operation).

Details of the implementation now follow. The basic systolic architecture (figure 2.2) consists of a linear array of compute cells, a routing network and a linear array of accumulator cells. The systolic array processor takes the bit-map of the binary image as input and computes the Hough transform for a particular value of θ.

Lemma 2.1: Suppose the image is scanned horizontally (vertically), that is one column (row) of pixels are sent into the systolic array per time unit as shown in figure 2.2. The destinations of the pixels for a given value of θ, say θ_k, can be computed by simple addition.

Proof: Let P_{ij} denote the pixel at (i,j) in the image plane. If the image is scanned horizontally, the row of pixels $P_{0j}, P_{1j}, ..., P_{N-1,j}$ will be sent to the j-th compute cell on successive time units. The destination r_{ij} of pixel P_{ij} is:

$$r_{ij} = (i\cos\theta_k + j\sin\theta_k) / \Delta\rho.$$

The corresponding destination of the next pixel $P_{i+1,j}$ that enters the j-th compute cell is

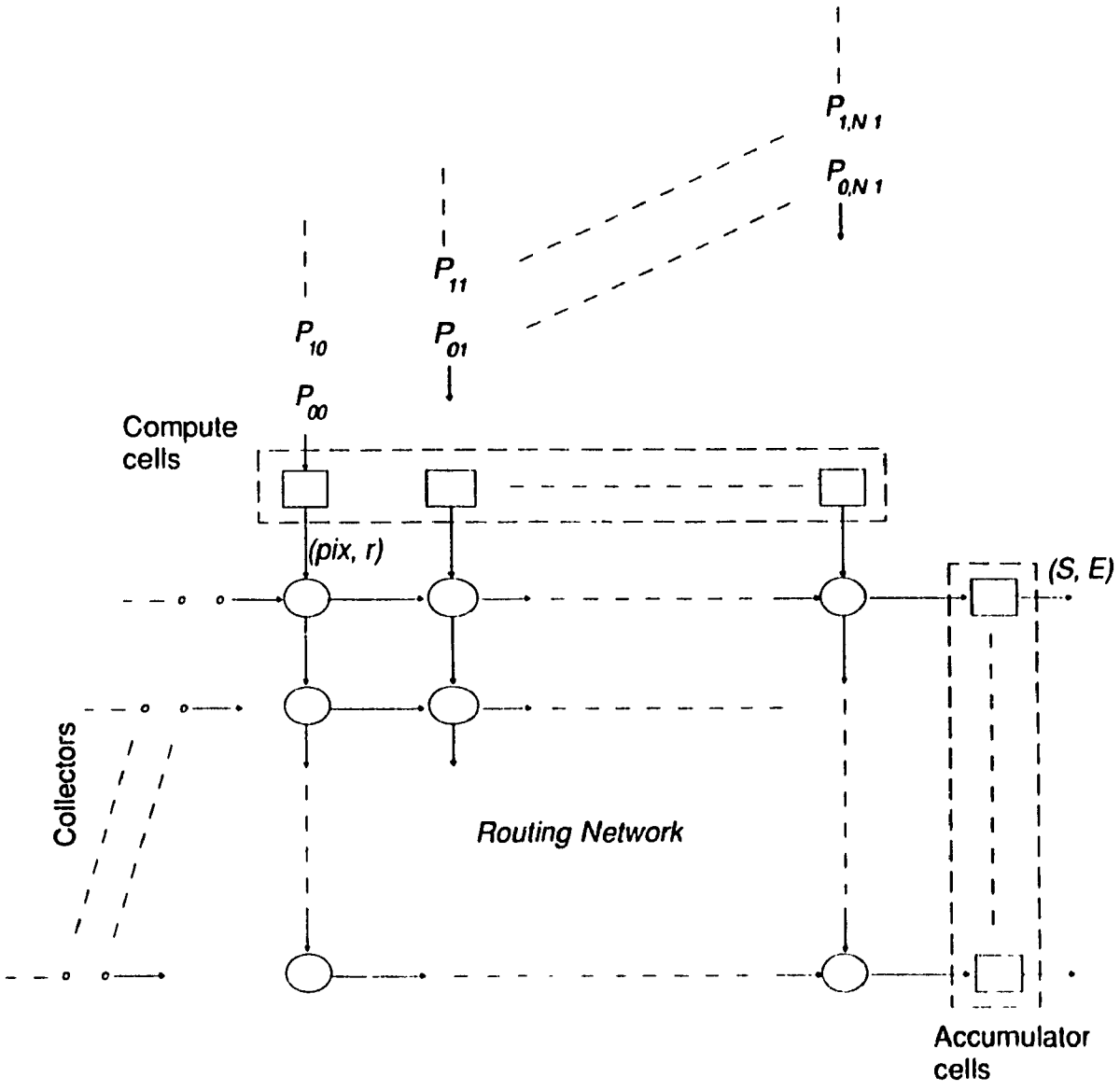


Figure 2.2 Architecture of the systolic array.

$$r_{i+1,j} = ((i+1)\cos\theta_k + j\sin\theta_k) / \Delta\rho.$$

The difference in destination between adjacent pixels that enter the same compute cell is a constant equal to

$$r_{i+1,j} - r_{ij} = \cos\theta_k / \Delta\rho.$$

By preloading the destination register with $j\sin\theta_k / \Delta\rho$, the computation of the destination of an edge pixel can be performed using simple addition. The value of $\cos\theta_k / \Delta\rho$ is added to the destination register in every time unit. When P_{ij} enters the j-th compute cell on the i-th time unit, the value of the destination register will be $(i\cos\theta_k + j\sin\theta_k) / \Delta\rho$. The incoming pixel can simply pick up the value in the destination register. If the image is scanned vertically, then the destination register of the i-th compute cell is preloaded with $i\cos\theta_k / \Delta\rho$. Destinations of two adjacent pixels differ by $\sin\theta_k / \Delta\rho$. □

The pseudo code of the compute cell function is shown below. Here, input and output variables are named with the suffix *_in* and *_out*, respectively. The variables without the suffix *_in* or *_out* are local variables. Since the destination of a pixel is within the range $[-m_\rho/2, m_\rho/2]$, it can be made always positive by adding an offset of $m_\rho/2$ to it. Hence the destination register is initialized to $j\sin\theta_k / \Delta\rho + m_\rho/2$.

In the following discussion, we assume that the image is scanned horizontally. The output of the compute cell is a two-tuple (pix, r), where pix is the value of the pixel and r is its destination. The routing network is responsible for sending the '1' pixels with destination r to the r-th accumulator cell. The routing network is composed of a $m_\rho \times N$ array of routing cells. The output of a compute cell (pix, r) traverses the routing network from top to bottom. The value of r is decremented by one whenever the output moves one row downward. When r becomes zero, the pixel will be picked up by a data token, called the "collector", that traverses the routing network

Function of the compute cell

/* input variables are named with suffix *_in*;
output variables are named with suffix *_out*;
variables without the suffix *_in* or *_out* are local variables;
*/

Data types:

pix : 1 bit;
r : integer;
increment, destination : fixed point number;

Initialization of the *j*-th compute cell:

increment $\leftarrow \cos\theta / \Delta\rho$;
destination $\leftarrow j\sin\theta / \Delta\rho + m_\rho/2$.

Computation:

pix_out $\leftarrow$ *pix_in*;
r_out $\leftarrow$ round(*destination*);
destination $\leftarrow$ *destination* + *increment*;

from left to right.

A collector consists of a bit-vector (shift register) of k bits, a modulo k counter, and two boolean flags, *pick_up* and *freeze_count*. The bit-vector is initialized to null (all zeros), the value of *count* is initialized to zero, and the two boolean flags are initialized to false. The collector has its count value incremented by one when it moves from one routing cell to another. When the collector first meets a '1' pixel with destination $r = 0$, it starts the "pickup" process and the incrementing of the counter is stopped. The value of the counter is used by the accumulator cell to compute the offset of the bit-vector with respect to the reference location. This will be elaborated later in this section. If $r = 0$, the value of the pixel is shifted into the bit-vector; otherwise a '0' is shifted into the bit-vector. The shifting of the bit-vector is from say left to right. The pickup process stops when the bit-vector is full, that is when the rightmost bit has a value '1'. The function of the routing cell is shown below. The nesting of the IF-ELSE statement is understood by the indentation in the pseudo code. If the column of pixels are sent into the array of compute cells with a 45° skewing (as in

Function of the routing cell

/* variables are named as in the function of the compute cell */

Data types:

r : integer;
pix : 1 bit;
bit_vector : *k*-bit array;
count : 0..*k*−1;
pick_up, *freeze_count* : boolean;

Computation:

```
pix ← pix_in;
bit_vector ← bit_vector_in;
freeze_count ← freeze_count_in;
pick_up ← pick_up_in;

IF (freeze_count = False)
   count ← (count_in + 1) mod k;
ELSE
   count ← count_in;

r ← r_in − 1;
IF (r = 0)
   IF (pix = 1)                    /* pick up the edge pixel     */
      shift pix into bit_vector;   /* stop incrementing counter */
      freeze_count ← True;
      pick_up ← True;
   ELSE
      IF (pick_up = True)
         shift '0' into bit_vector;
   pix_out ← 0;
ELSE
   IF (pick_up = True)
      shift '0' into bit_vector;
   pix_out ← pix;

IF (bit_vector is full)  /* the rightmost bit = 1    */
   pick_up ← False;  /* stop the pick up process */

r_out ← r;
count_out ← count;
bit_vector_out ← bit_vector;
freeze_count_out ← freeze_count;
pick_up_out ← pick_up;
```

figure 2.2), then the pixels on the j-column will meet the j-th collector at successive routing cells on each row of the routing network. Pixels with destination equal to r will be picked up by the corresponding collector on the r-th row counting from the top. The size of the bit-vector is given by the following lemma.

Lemma 2.2: If we scan the image horizontally (process one column of pixels per time unit) for $45° \le \theta \le 135°$ and vertically (process one row of pixels per time unit) for $0° \le \theta < 45°$ and $135° < \theta < 180°$, then k, the number of pixels on the same column (row) having the same destination, will be bounded by $\sqrt{2} \times \Delta\rho$.

Proof: Consider the case of horizontal scanning. Referring to figure 2.3, the value of k is:

$$k = \Delta\rho \,/\, \sin\theta.$$

For $45° \le \theta \le 135°$, $\sin\theta > 1/\sqrt{2}$. Similarly, for the case of vertical scanning,

$$k = \Delta\rho \,/\, \cos\theta \quad \text{for } 0° < \theta < 45° \text{ and}$$

$$k = \Delta\rho \,/\, \cos(180°-\theta) \quad \text{for } 135° < \theta < 180°.$$

For the given ranges of θ, $\cos\theta > 1/\sqrt{2}$. □

The discretization error in the Hough transform has been studied by van Veen and Groen [Van81]. The optimum values of $\Delta\rho$ and $\Delta\theta$ are related by

$$\Delta\rho = L_{max} \times \sin(\Delta\theta/2)$$

where L_{max} is the maximum length of a line in the image. Value of $\Delta\rho$ should be larger than 1 to avoid oversampling. To obtain a good accuracy, value of $\Delta\rho$ should not be too large. In this thesis, we choose $\Delta\rho$ to be less than or equal to three units, and $\Delta\theta$ to be 1°.

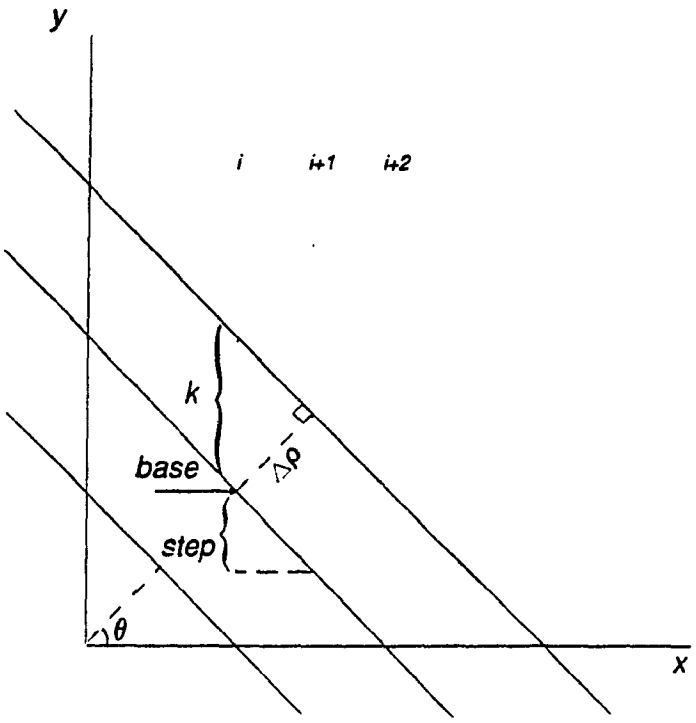


Figure 2.3 Parameters used in the accumulator cell with horizontal scanning: $k = \Delta\rho / \sin\theta$ and *step* $= 1/\tan\theta$.

The pseudo code of the accumulator cell function is shown below. The accumulator cell is "triggered" by the arrival of the first collector. The value of the counter *current_cycle* corresponds to the column number, that is the x–coordinate of the pixels in the incoming bit-vector. The accumulator cell keeps in itself an expected value of the count (*ref_count*) that corresponds to a reference location (*base*) as shown in figure 2.3. By comparing the value of *ref_count* and the value of *count* of the collector, the offset of the bit-vector can be obtained. Based on the value of the offset and the reference location stored in the base register, the coordinates of the pixel can be computed. In practice, only the coordinates of the start and end points need to be computed. Knowing the offset, the bit-vector can then be aligned and compared with the previous bit-vector (*last_bit_vector*) in the bucket to check for pixel connectivity. Only the connected pixels will be retained and stored back to *last_bit_vector*. If a

Function of the accumulator cell

/* variables are named as in the function of the compute cell */

Data types:

bucket_empty : boolean;
count_in, ref_count, offset : 0..$k-1$;
base, step : fixed point number;
start, end, current_cycle, start_cycle : integer;
bit_vector_in, current_bit_vector, last_bit_vector : k-bit array;

Initialization:

preload *base*, *ref_count*, *step*;
current_cycle ← 0;
bucket_empty ← True;

Computation on receiving a collector:

offset ← *count_in* − *ref_count*;
current_bit_vector ← align(*bit_vector_in*, *offset*);
IF (*current_bit_vector* is empty)
  IF (*bucket_empty* = False)
    output (*start_cycle*, *start*) and (*current_cycle*−1, *end*);
      /* the start and end points of the line segment */
    bucket_empty ← True;
ELSE
  IF (*bucket_empty* = True)
    start ← round(*base*) + *offset*;
    end ← *start*;
    start_cycle ← *current_cycle*;
    last_bit_vector ← *current_bit_vector*;
    bucket_empty ← False;
  ELSE
    IF no pixel in *current_bit_vector* connected to pixels in *last_bit_vector*
      output (*start_cycle*, *start*) and (*current_cycle*−1, *end*);
        /* the start and end points of the line segment */
      last_bit_vector ← *current_bit_vector*;
      cycle_start ← *current_cycle*;
      start ← round(*base*) + *offset*;
      end ← *start*;
    ELSE
      last_bit_vector ← connected pixels in *current_bit_vector*;
      end ← round(*base*) + offset of the last connected pixels;

current_cycle ← *current_cycle* + 1;
base ← *base* + *step*;
ref_count ← round(*base*) mod k;

disconnectivity is detected (incoming bit-vector empty or no connected pixel found), the current line segment (if exists) will be output, and the incoming bit-vector (if it is not empty) defines the starting of a new line segment. The values of *base* and *ref_count* are updated during every time unit by adding to them the value of *step*.

Lemma 2.3: An accumulator cell groups connected edge pixels into a line segment defined by the start and end points.

The proof of lemma 2.3 follows directly from the above description. It is important to note that *step*, *base*, and *ref_count* are real (fixed point) numbers. However, only the integer part of *base* and *ref_count* will be used in the computation of the offset, pixel coordinates, etc.

Theorem 2.1: The systolic array correctly computes the straight line Hough Transform of a binary image such that each detected line segment is composed of a string of connected pixels.

The proof of theorem 2.1 follows directly from the above discussion. Each line segment detected is contiguous and is defined by the four-tuple (θ,ρ,S,E), where S and E are the start and end points of the line segment, respectively.

2.1.1. A Remark on Systolic Computation

Systolic arrays evolved as an architecture for direct VLSI implementation which emphasizes the regularity and locality of communications. These two properties are, in general, the basic prerequisites for efficient parallel processing. From this viewpoint, systolic computation can be regarded as a general parallel processing paradigm. Systolic algorithms are specifications of parallel computations (software architectures) which can be embedded on various forms of parallel machines, such as SIMD array or tightly coupled (shared memory) MIMD multiprocessors systems [Hwa84]. Embedding

data flow systolic algorithms onto SIMD arrays is straightforward. Considering the design of the Geometric Arithmetic Parallel Processor (GAPP) [Dav84], which is a bit-serial systolic array, the major difference between data flow systolic computations and SIMD computations lies mainly on the initial data distribution.

In MIMD multiprocessors systems, bottlenecks of computation usually arises from memory contention and delays due to irregular synchronization among processors. From the point of view of software architecture, each data path in a systolic array is a thread of control that forms a natural software pipeline of subtasks. Each pipeline receives (feeds) data tokens from (to) adjacent pipelines synchronously. When implementing systolic algorithms in MIMD machines, the algorithm is decomposed into a number of software pipelines and each pipeline is run on a separate processor. Both memory contention and synchronization delays can be avoided. Optimal speedup proportional to the number of processors in the system is achievable.

Systematic methodology for designing systolic arrays has been a popular research topic in recent years. Methods to generate systolic architectures from mathematical specifications, such as set of linear recurrence equations, have been developed [Mol82, KuSY88]. Research in hardware algorithm design may as well be applied to parallel programming and compiler design.

2.2. Area-Time Complexity

This section analyzes the area-time (AT) complexity of the architecture presented in the previous section. The AT complexity of the proposed architecture is compared to that of the conventional sequential algorithm and some parallel designs. The area costs of the three types of processing cells in our design are first considered.

1. *Compute cell:* If the image size is $N \times N$, then the destination register should have $O(\log N)$ bits (we assume that m_ρ has the same order of N). It has been shown in [Dud75] that the allowable truncation error in $\cos\theta$ and $\sin\theta$ is $\Delta\rho/N$. If $\Delta\rho > 1$,

then $\log N$ bits are required to store the value of $\cos\theta$ or $\sin\theta$. Hence the area cost of the compute cell is $A_c = O(\log N)$.

2. *Routing cell:* For values of $\Delta\rho$ less than or equal to some constant, the length of the bit vector is bounded. So the area of the routing cell is dominated by the register storing the destination r, which is $A_r = O(\log N)$.

3. *Accumulator cell:* In the accumulator cell, there are registers storing the start and end points of a line segment. The bit vector and comparator have constant area. Hence, the area cost of the accumulator cell is $A_a = O(\log N)$.

Thus, the total area of the systolic array is:

$$\begin{aligned} A_{Pao} &= N \times A_c + m_\rho \times N \times A_r + m_\rho \times A_a \\ &= O((N + m_\rho N + m_\rho)\log N) \\ &= O(m_\rho N \log N). \end{aligned}$$

Assume that the arithmetic and logical operations can be done in one time unit. The time required to scan the image once is N units. To compute the transform, the image have to be scanned m_θ times or m_θ systolic arrays are required if these are to be done concurrently. Thus the AT complexity of our architecture is:

$$AT_{Pao} = O(m_\rho m_\theta N^2 \log N).$$

The AT complexity of the conventional sequential algorithm [Han87] can be abstracted as follows. The area cost consists of two components: (1) the storage space, and (2) space for the computation unit. For images of reasonable sizes, the total area cost is dominated by the storage cost. Each accumulator cell records the number of votes it got. Hence, the area cost of the accumulator cell is $O(\log N)$.

$$A_{seq} = O(m_\rho m_\theta \log N).$$

The total time required is the sum of the computation time of the following two processes.

1. Scanning the $N \times N$ image to obtain the coordinates of the edge pixels. This process requires $O(N^2)$ time units.
2. Computing for each edge pixel the value of ρ for each possible value of θ. This process requires $O(P\, m_\theta)$ time units, where P is the number of edge pixels in the image.

However, the two computation processes can be pipelined. For reasonably sized images, the computation time is dominated by the second process. Thus, the AT complexity of the sequential algorithm is:

$$AT_{seq} = O(P\, m_\rho m_\theta^2 \log N).$$

Usually, P is equal to a few percent of the image area. For large N, $P\, m_\theta > N^2$. Hence, our architecture has better AT complexity.

Chuang et al [Chu85] designed a systolic array processor based on Shu's approach [Shu87]. Instead of using a bit array, Chuang stores the distances of the pixels with respect to a reference point (the first pixel mapped to the bucket) in the accumulator cell. Connectivity check is then accomplished by first sorting the "distances", and a big gap between two consecutive distances indicates a discontinuity. Hence, the area cost of the accumulator array is:

$$A_{Chuang} = O(m_\rho m_\theta N \log N).$$

The time required to compute the transform is $O(P + m_\theta + m_\rho)$ excluding the post-processing time (time for the sorting process). Hence the AT complexity of Chuang's architecture is:

$$AT_{Chuang} = O((P + m_\theta + m_\rho) m_\rho m_\theta N \log N).$$

The AT complexity of our architecture is superior to that of Chuang's since $P \gg N$ for reasonable size images.

During the course of our research, more parallel designs were being reported. Fisher and Highnam implemented the Hough transform on a linear array of SIMD processors called the scan line array processor (SLAP) [Fis89]. As the SLAP sweeps through the image plane from top to bottom, the PEs trace along lines with $45° \le \theta \le 135°$ and accumulate the votes. For other values of θ, the SLAP sweeps through the image from left to right. The AT complexity of this method is

$$AT_{SLAP} = O(m_\theta^2 N^2 \log N)$$

which is basically equal to that of our design.

Guerra and Hambrusch implemented the Hough transform on a 2D SIMD array [Gue89]. Assume the accumulator array is also of size $N \times N$. The input image is first preloaded onto the processor array with one pixel per PE. Two algorithms based on line tracing were developed. The first one partitions the $N \times N$ array into N blocks of size $\sqrt{N} \times \sqrt{N}$. Each block B_i is assigned a value of θ_i and each PE within a block is assigned a value of ρ_j, for $j = 1,2,...,N$. In one iteration, the PEs in a block traces along the lines with $\theta = \theta_i$ within the block B_i by sending message packets to adjacent PEs, and accumulate the votes. This process takes $O(\sqrt{N})$ time. Also, N iterations are required to compute the transform. Hence, the time complexity of the algorithm is $O(N\sqrt{N} + N)$. Hence the AT complexity is

$$AT_{Guerra1} = O(N^{3.5} \log N).$$

In the second one, no partitioning is required. The lines with $45° \le \theta \le 135°$ are traced from left to right across the 2D array and lines with other values of θ's are traced from top to bottom. Collision can be avoided if the tracing is done according to a certain order. The time complexity of the algorithm is $O(N+m_\theta)$. Hence, the AT complexity is

$$AT_{Guerra2} = O(N^3 \log N).$$

This is by far the asymptotically fastest algorithm reported in the literature. This method, however, requires substantial multiplication operations. Hence, the constant factor of the AT complexity is much larger than that of our method. The algorithm reported in [Cyp87] also exhibits $O(N)$ time complexity. The AT complexities of the six methods are summarized in table 2.1.

Table 2.1 Area-time complexities of the six Hough transform algorithms*

Methods	Area complexity	Time complexity	Area-Time complexity
Sequential	$O(m_\rho m_\theta \log N)$	$O(P m_\theta)$	$O(P m_\rho m_\theta^2 \log N)$
Chuang	$O(m_\rho m_\theta N \log N)$	$O(P + m_\theta + m_\rho)$	$O(P m_\rho m_\theta N \log N)$
SLAP	$O(m_\theta N \log N)$	$O(m_\theta N)$	$O(m_\theta^2 N^2 \log N)$
Guerra$_1$	$O(N^2 \log N)$	$O(N^{1.5})$	$O(N^{3.5} \log N)$
Guerra$_2$	$O(N^2 \log N)$	$O(N)$	$O(N^3 \log N)$
Pao	$O(m_\rho N \log N)$	$O(m_\theta N)$	$O(m_\rho m_\theta N^2 \log N)$

*In general, $P \gg N \approx m_\rho > m_\theta$.

Although the algorithms of Guerra and that of Cypher have a better AT complexity than ours, their algorithms cannot be extended to incorporate the pixel connectivity check.

2.3 Line Selection

Since a bucket can hold multiple collinear line segments, simple thresholding of the accumulator array is not applicable. In this section, we present a line selection algorithm that is based on local comparison among adjacent buckets. The Hough

transform can be regarded as a sampling tool which samples the input image at different orientations to collect evidence of the presence of straight lines. Conversely, a straight line in the image will be sampled by more than one buckets in the θ–ρ space. Each detected line segment whose length is larger than $L_{\min}$ is a sampled instance of the line. We want to select one of the sampled instances as the representative. The longest line segment that passes through the set of pixels of the line is selected. However, we don't know in advance what this set is. A heuristic procedure is adapted and described below.

A line segment is defined by the four-tuple (θ,ρ,S,E), where S and E are the starting and ending points of the line, respectively. Let $L_1 = (\theta_m, \rho_k, S_1, E_1)$ be a line segment detected by bucket (θ_m, ρ_k). Some portion of L_1 will map into (vote for) buckets in adjacent columns $(\theta_{m+\delta}, \rho_{k'})$ as shown in figure 2.4. In fact, the votes for L_1 can spread over $2\delta+1$ columns in the accumulator array, where δ is given by the following lemma.

Lemma 2.4a: If $L_{\min}$ is the minimum length among the lines of interest in an image, then the spreading of votes of any of these lines are confined to $2\delta+1$ columns in the accumulator array, where

$$\delta = \Delta\rho \,/\, (\Delta\theta \times L_{\min}).$$

Proof: Referring to figure 2.4, the line L_1 is detected by bucket (θ_m, ρ_k). The portion of L_1, namely L_{ab}, that also votes for bucket $(\theta_{m+\delta}, \rho_{k'})$ is

$$L_{ab} = \Delta\rho \,/\, \sin\alpha > L_{\min},$$

where $\alpha = \delta \times \Delta\theta$. Hence,

$$\Delta\rho \,/\, L_{\min} > \sin(\delta \times \Delta\theta) \approx \delta \times \Delta\theta$$

implying that, $\delta < \Delta\rho \,/\, (\Delta\theta \times L_{\min})$. □

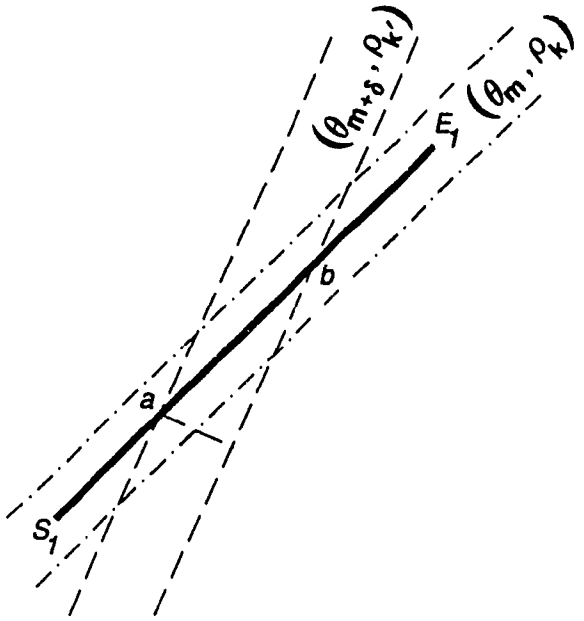


Figure 2.4 Spreading of a line segment to buckets in adjacent columns.

Lemma 2.4b: The spreading of votes of line $L_1 = (\theta_m, \rho_k, S_1, E_1)$ in column $\theta_{m'}$ of the accumulator array is bound by the range $[\min(R_s, R_e), max(R_s, R_e)]$, where

$$R_s = (S_1(x)\cos\theta_{m'} + S_1(y)\sin\theta_{m'}) / \Delta\rho,$$

and

$$R_e = (E_1(x)\cos\theta_{m'} + E_1(y)\sin\theta_{m'}) / \Delta\rho,$$

where $S(x)$, $S(y)$ and $E(x)$, $E(y)$ denote the $x-$ and $y-$coordinates of the start and end points, respectively.

The proof of lemma 2.4b is straightforward, and is omitted. When we compare two lines, we have to, first of all, be sure that the two lines correspond to the same group of points in the image within the specified tolerance. Two lines $L_1 = (\theta_1, \rho_1, S_1, E_1)$ and $L_2=(\theta_2, \rho_2, S_2, E_2)$ as depicted in figure 2.5 are said to

correspond to the same group of points in the image if:

1. L_1 intersects or touches L_2, and
2. the perpendicular distances of the start and end points of L_2 to L_1, say D_s and D_e, are both smaller than some value, say $\Delta\rho$+1. The values of D_s and D_e are computed as follow:

$$D_s = S_2(x)\cos\theta_1 + S_2(y)\sin\theta_1,$$

$$D_e = E_2(x)\cos\theta_1 + E_2(y)\sin\theta_1.$$

In fact, a bounding box of width $2(\Delta\rho+1)$ and length L_1 is defined. The portion of L_2 that falls within this bounding box is regarded as overlapping with L_1. If the length of L_1 is longer than that of L_2 and significant portion of L_2 overlaps with L_1 (say 75%), then L_1 is selected and L_2 is discarded. The approximate percentage overlap is calculated as follows:

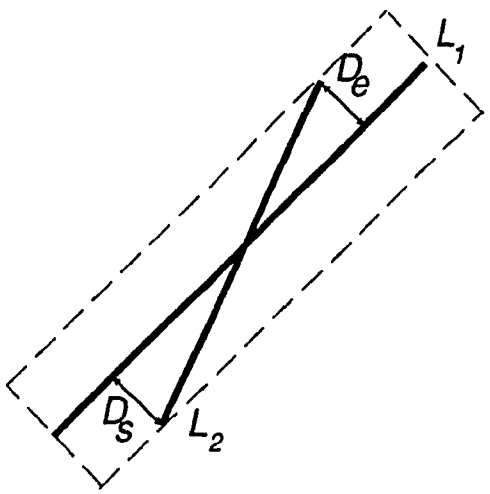


Figure 2.5 Checking for overlapping lines.

Case 1: $45° \le \theta \le 135°$

$$P_1 = \max(S_1(x), S_2(x)); \quad P_2 = \min(E_1(x), E_2(x));$$

$$\%\,\text{overlap} = \frac{P_2 - P_1}{E_2(x) - E_1(x)} \times 100.$$

Case 2: $0° \le \theta < 45°$ and $135° < \theta < 180°$

$$P_1 = \max(S_1(y), S_2(y)); \quad P_2 = \min(E_1(y), E_2(y));$$

$$\%\,\text{overlap} = \frac{P_2 - P_1}{E_2(y) - E_1(y)} \times 100.$$

The pseudo code of the line selection algorithm is shown below. It is important to note that L_1 will not be marked when it serves as the initiator of the checking process. This makes the parallelization of the line selection process easy. Each bucket is simply required to broadcast its line segments to its neighbors whose range is given by lemmas 2.4a and 2.4b. All line segments that are not marked will be selected after the process terminates.

Line Selection Algorithm1.

```
FOR each line segment L1=(θ1,ρ1,S,E)
   FOR θ2 = θ1–δ to θ1+δ
      rs = S(x)cosθ2 + S(y)sinθ2;
      re=E(x)cosθ2 + E(y)sinθ2;
      FOR r2 = rs to re
         FOR all L2 in bucket (θ2,ρ2)
            check_overlap(L1,L2)
            IF 75% of L2 overlap with L1 and length of L2 < length of L1
               mark L2;

select all line segments not marked;
```

2.4. Experimentation

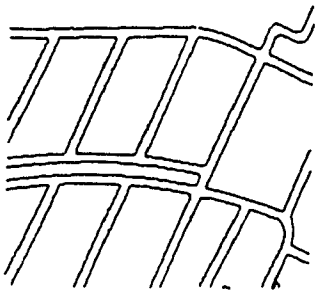
The modified Hough transform and the systolic architecture presented in section 2.1 were simulated to evaluate the effect of considering pixel contiguity. The test image (figure 2.6a) was extracted from a road map and digitized with a resolution of 200 points per inch. With this resolution, the width of a road is about 8 pixels. Figure 2.6b shows the edges extracted from this image after preprocessing. The values of $\Delta\rho$ and $\Delta\theta$ are 3 units and 2 degrees, respectively. Line segments with length less than 10 units are discarded. Each line segment is counter-checked with buckets that are less than or equal to 5 columns apart. The results of the modified transform is shown in figure 2.6c and figure 2.6d is the image reconstructed from the selected line segments. The curved contours in the image are approximated by a number of straight line segments. Several short edges marked in figure 2.6d are missing. If the image was digitized with a higher resolution, edge A would have been detected. The road contour near edge B is slightly curved. The line segment that contains edge B was removed because more than 75% of its length overlaps with the adjacent line segment and it is shorter. The other missing edges are removed due to the same reason.

2.5. Remarks

Logic circuits of the processing cells of the proposed systolic architecture have been designed. The number of logic gates required to implement the compute cell, routing cell and accumulating cell are 288, 209 and 817, respectively. The average number of transistors per gate is about 4 to 5 if implemented using CMOS technology. With today's technology, more than a hundred of such processing cells can be fabricated on a single chip.

The modified Hough transform was simulated and the observed performance is satisfactory. The accuracy of the modified transform in describing the original image is remarkable. In the line selection algorithm, a line segment L in bucket (θ,ρ) is

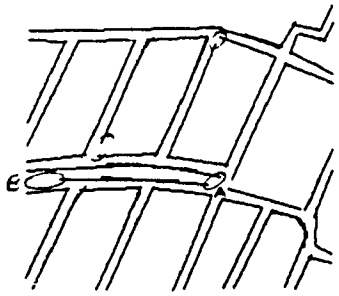
(a)



(b)

```
total number of line segments detected   = 1831
number of line segments selected         =   72
length of longest line segment selected  =  145 units
length of shortest line segment selected =   14 units
```

(c)



(d)

Figure 2.6 Example of straight lines detection: (a) Test image extracted from a road map; (b) edge pixels extracted after preprocessing; (c) results of the modified HT; and (d) image reconstructed from the selected line segments.

removed if (1) significant portion of L overlaps with another line segment L' that belongs to an adjacent bucket, and (2) L is shorter than L'. If this line selection algorithm is modified slightly such that only the portion of L that overlaps with L' is removed, a better linearization of the image can be obtained.

Basically, the line selection can be done in a number of ways, depending on the objectives and requirements of the applications. The algorithm presented in section 2.4 will extract an approximate linearization of the image. The extracted features will be useful in applications such as character recognition [Che89, Kus85], polygonal object recognition [Eng88] and vehicle guidance [Ini84] etc. Another line selection algorithm will be presented in chapter 6, where we are only interested in extracting the "true" long straight lines.

Recently, Princen, Illingworth and Kittler developed a bottom-up hierarchical approach for detection of straight lines [PIK90]. The input image is initially divided into a number of overlapping subimages. The Hough transform is applied to individual subimages with a coarse angular resolution. Adjacent subimages will be merged together to form a larger parent subimage in the next higher level. Each detected line segment is represented by the intersection point of the hypothetical line and its normal. The grouping of line segments in the parent subimage is again expressed as one of detecting collinear points. The angular resolution is refined as the grouping proceeds up the hierarchy. The grouping process is event driven and stops when (i) the vote count of the new line obtained is not larger than any of its contributors; or (ii) the top level is reached. This method essentially integrates the voting process and the line selection process into one process. Because of the initial partition of the input image and local grouping, distant noise pixels or pixels of far apart objects are not likely to contribute to the vote of the line. This is also a legitimate approach to improve the sensitivity of the Hough transform. However, the connectivity constraint becomes more relaxed when proceeding up the hierarchy. Line segments, which are represented

by points in adjacent subimages will be grouped together if the representative points satisfy the collinearity constraint. In the upper level of the hierarchy, the sizes of the subimages, as well as the allowable distances between line segments in the group, become larger. Therefore, it is still possible to group together collinear but separated line segments into one line.

Chapter 3

Characterization of 2D Smooth Curves Using the Set of Tangent Lines

Pattern recognition methodologies depend heavily on how one characterizes or represents the shape of an object. A brief survey of various shape representation schemes can be found in [Bal82, ch.8]. In this chapter, we propose a new representation scheme for 2D objects based on the set of tangent lines. It will be shown in section 3.1 that a simple (planar) closed smooth curve in the continuous domain can be uniquely characterized by its (complete) set of tangent lines; that is, two curves with the same set of tangents have to be identical.

This representation scheme by itself is of interest in the study of geometry, where the problem is known as the uniqueness problem [Hor89]. The motivation of our work, however, originated from the study of the Hough transform. Recognition methodologies based on the straight line Hough transform [Cas87, Kri87a] and the curve detection methods to be presented in chapters 4 and 5 recognize objects in the θ–ρ space. Each point in the θ–ρ space corresponds to a line in the x–y plane which is tangent to the curve. The results derived in this chapter will serve as the theoretical basis of these recognition methodologies.

3.1. The Shape Representation Scheme

Assumption 1 (Smoothness assumption): The curve C is continuous, smooth, simple and closed.

Assumption 2 (Finite tangency): Let $T(C)$ be the set of straight lines that are tangent to C. There are only finitely many lines in $T(C)$ that have the same inclination. In other words, in any given direction, there are only a finite number of lines that are tangent to C.

If $T(C)$ satisfies assumption 2, then we will be able to uniquely reconstruct C from $T(C)$ without knowing the point(s) of tangency on each tangent line (as shown later in this section). This implies that if two curves have the same set of tangents, then they have to be identical. Some previous studies on shape reconstruction from tangent lines (supporting lines) are restricted to reconstruction of the convex hull [Mur88, Pri90].

3.1.1. Preliminaries and Notations

A curve C in the $x-y$ plane can be expressed as

$$C = C(s) = (x(s), y(s))$$

where s is the arc length measured along C in the counterclockwise direction from a reference point o (which is an arbitrary point on C). Thus, $(x(s), y(s))$ define a moving point in the $x-y$ plane when the curve C is traversed. If C is smooth, then the first derivatives $x'(s)$ and $y'(s)$ exist and will not be both equal to zero for any s.

Let the line L_s be tangent to C at the point $(x(s), y(s))$. The slope of L_s is

$$m_s = \tan\phi(s) = \frac{y'(s)}{x'(s)}$$

where $\phi(s)$ is the angle measured from the positive $x-$axis to the line L_s. The range of $\phi(s)$ is from 0° to 180°. The line L_s can also be characterized by $(\theta(s), \rho(s))$:

$$L_s : \rho(s) = x\ \cos\theta(s) + y\ \sin\theta(s)$$

where $\rho(s)$ is the perpendicular distance of the origin from L_s and $\theta(s)$ is the angle

measured from the positive x–axis to the line normal to L_s. We have the slope of the normal

$$\tan\theta(s) = \frac{-x'(s)}{y'(s)} .$$

We can also observe that $\phi(s)$ and $\theta(s)$ are related by

$$\theta(s) = (\phi(s) + 90°) \bmod 180, \text{ and}$$

$$\phi(s) = (\theta(s) + 90°) \bmod 180.$$

The pair $(\theta(s), \rho(s))$ defines a corresponding moving point in the θ–ρ space[2].

Definition

Transformation H: Suppose we traverse the curve C in counterclockwise direction, and compute the tangent line to C at $(x(s),y(s))$ and the corresponding moving point $(\theta(s), \rho(s))$ in the θ–ρ space. The resultant *pattern* traced by the moving point $(\theta(s), \rho(s))$ is denoted as $H(C)$.

Although $H(C)$ is obtained from the trace of $(\theta(s), \rho(s))$, it is represented by the set of points visited by $(\theta(s), \rho(s))$. Figure 3.1 depicts a simple closed smooth curve and the transformed pattern.

3.1.2. Main Theorem

Theorem 3.1: For two curves C_1 and C_2 that satisfy assumptions 1 and 2, $H(C_1) = H(C_2)$ if and only if $C_1 = C_2$.

We will prove the theorem by showing that given $H(C)$, the original curve C can be uniquely reconstructed. First, we consider some of the properties of $H(C)$.

[2] Every straight line in the x–y plane is uniquely represented by a point in the θ–ρ space, except for vertical lines. A vertical line, $x = a$, is mapped to two points $(0°,a)$ and $(180°,-a)$ in θ–ρ space.

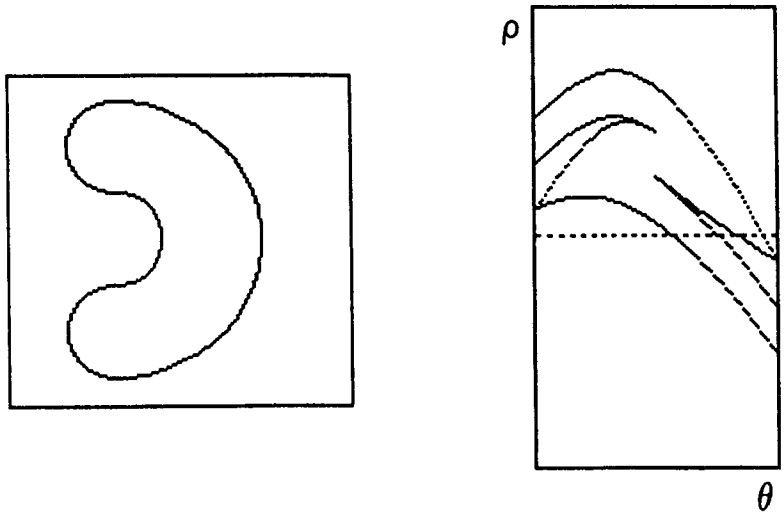


Figure 3.1 A simple closed smooth curve and its H transform.

Lemma 3.1: If C satisfies assumption 1, then

(i) the trace of the moving point $(\theta(s), \rho(s))$ in the θ–ρ space is continuous, except that there is a discontinuity jump from some point $(180°, -a)$ to the point $(0°, a)$, and vice versa; and

(ii) $(\theta(s), \rho(s))$ will not move vertically.

Proof: Recall that $\tan\theta(s) = -x'(s)/y'(s)$, and $\rho(s) = x(s)\cos\theta(s) + y(s)\sin\theta(s)$. Since C is continuous and smooth,

$$\lim_{s_1 \to s_0} x'(s_1) \to x'(s_0), \text{ and } \lim_{s_1 \to s_0} y'(s_1) \to y'(s_0).$$

To show (i), we consider the following three cases separately.

(a) $y'(s_0) \neq 0$ and $x'(s_0) \neq 0$.

For any point s_1, it is easy to observe that

$$\lim_{s_1 \to s_0} \theta(s_1) \to \theta(s_0), \text{ and } \lim_{s_1 \to s_0} \rho(s_1) \to \rho(s_0).$$

(b) $y'(s_0) \neq 0$ and $x'(s_0) = 0$.

In this case, the tangent is a vertical line. This line is mapped to the two points $(0°, x(s_0))$ and $(180°, -x(s_0))$ in θ–ρ.

For s_1 such that $x'(s_1) > 0$

$$\lim_{s_1 \to s_0} \theta(s_1) \to 180° \text{ and } \lim_{s_1 \to s_0} \rho(s_1) \to -x(s_0) \quad \text{if } y'(s_0) > 0,$$

$$\lim_{s_1 \to s_0} \theta(s_1) \to 0° \text{ and } \lim_{s_1 \to s_0} \rho(s_1) \to x(s_0) \quad \text{if } y'(s_0) < 0.$$

For s_1 such that $x'(s_1) < 0$

$$\lim_{s_1 \to s_0} \theta(s_1) \to 0° \text{ and } \lim_{s_1 \to s_0} \rho(s_1) \to x(s_0) \quad \text{if } y'(s_0) > 0,$$

$$\lim_{s_1 \to s_0} \theta(s_1) \to 180° \text{ and } \lim_{s_1 \to s_0} \rho(s_1) \to -x(s_0) \quad \text{if } y'(s_0) < 0.$$

(c) $y'(s_0) = 0$ and $x'(s_0) \neq 0$.

In this case the tangent is a horizontal line which is mapped to $(90°, y(s_0))$.

For s_1 such that $y'(s_1) > 0$,

$$\lim_{s_1 \to s_0} \theta(s_1) \to 90°^{+} \quad \text{if } x'(s_0) > 0,$$

$$\lim_{s_1 \to s_0} \theta(s_1) \to 90°^{-} \quad \text{if } x'(s_0) < 0.$$

For s_1 such that $y'(s_1) < 0$,

$$\lim_{s_1 \to s_0} \theta(s_1) \to 90°^{-} \quad \text{if } x'(s_0) > 0,$$

$$\lim_{s_1 \to s_0} \theta(s_1) \to 90°^{+} \quad \text{if } x'(s_0) < 0.$$

In both cases, $\lim_{s_1 \to s_0} \rho(s_1) \to y(s_0)$.

To show (ii), we will derive a contradiction. Without loss of generality assume that $(\theta(s), \rho(s))$ moves vertically upward (or downward) at $\theta = 90°$ for some $s \in [s_0, s_1]$. Recall that $\tan\theta(s) = -x'(s)/y'(s)$. If $\theta(s) = 90°$, then $y'(s) = 0$. This implies that the corresponding curve segment $C(s)$ for $s \in [s_0, s_1]$ should be a horizontal line. Hence $\rho(s)$ cannot move upward (or downward) for $s \in [s_0, s_1]$. □

Before we present the reconstruction of general smooth closed curves, we first consider the reconstruction of simple open convex (or concave) curves. Let $F = F(x)$ for $x \in [a,b]$ be a single valued function[3] which is continuous and smooth.

Lemma 3.2: If $H(F)$ is a single continuous curve in θ–ρ such that (i) $0° \le \theta \le 180°$, and (ii) no two points in $H(F)$ possess the same θ value, then F is either convex or concave.

Proof: $H(F)$ is the set of points visited by the moving point $(\theta(s), \rho(s))$ when F is traversed. In this particular case, since no two points in $H(F)$ can have the same θ value, $\theta(s)$ must be monotonic in s. Suppose $\theta(s)$ is monotonically increasing in s. Since $\phi = (\theta + 90°) \bmod 180$,

$$\lim_{\theta \to 0^+} \phi \to 90^{o+} \Rightarrow \tan\phi \to -\infty$$

$$\lim_{\theta \to 90°} \phi \to 0° \Rightarrow \tan\phi \to 0$$

$$\lim_{\theta \to 180^{o-}} \phi \to 90^{o-} \Rightarrow \tan\phi \to +\infty.$$

Hence, $F'(x)$ is nondecreasing in $a \le x \le b$. Therefore, $F(x)$ is convex [Roc70, Thm. 4.4]. Similarly, if $\theta(s)$ is monotonically decreasing in s, then $F(x)$ is concave. □

In the following discussion, F is assumed to be either convex or concave.

Definitions

Let $(a, F(a))$ and $(b, F(b))$ be the two end points of F. Let L_a and L_b be the two tangent lines to F at its two end points, respectively. The *extended curve* F_e is defined as follows:

[3] The terms "curve" and "function" are used synonymously in this chapter.

$$F_e = \begin{cases} L_a^+ & \text{for } x \le a \\ F & \text{for } a < x < b \\ L_b^+ & \text{for } x \ge b \end{cases}$$

where

$$L_a^+ = \begin{cases} L_a & \text{if } L_a \textit{ is non-vertical} \\ \{(x,y) \mid (x,y) \in L_a \text{ and } y \ge F(a)\} & \text{if } L_a \textit{ is vertical and } F \textit{ is convex} \\ \{(x,y) \mid (x,y) \in L_a \text{ and } y \le F(a)\} & \text{if } L_a \textit{ is vertical and } F \textit{ is concave} \end{cases}$$

and L_b^+ is defined similarly. The *domain* of F_e denoted as Dom_F_e is defined as follows:

$$Dom\_F_e = \begin{cases} (-\infty, +\infty) & \text{if } L_a \text{ and } L_b \textit{ are non-vertical} \\ (-\infty, b] & \text{if } L_a \textit{ is non-vertical and } L_b \textit{ is vertical} \\ [a, +\infty) & \text{if } L_a \textit{ is vertical and } L_b \textit{ is non-vertical} \\ [a, b] & \text{if } L_a \text{ and } L_b \textit{ are vertical} \end{cases}$$

The *epigraph* of F_e denoted as epi_F_e is defined as follows:

$$epi\_F_e = \begin{cases} \{(x,y) \mid x \in Dom\_F_e \text{ and } y \ge F_e(x)\} & \text{if } F \textit{ is convex} \\ \{(x,y) \mid x \in Dom\_F_e \text{ and } y \le F_e(x)\} & \text{if } F \textit{ is concave} \end{cases}$$

Hence, the boundary of $epi\_F_e \triangleq F_e$. Let a' and b' be two x's such that

(i) $a' \ge a$ and $b' \le b$, and

(ii) the point $(a', F(a'))$ lies on L_a and $(b', F(b'))$ lies on L_b, and

(iii) for $x \in (a', b')$, $(x, F(x))$ does not lie on either L_a or L_b.

Then the *reduced curve* F_r is defined as

$$F_r = F \text{ for } x \in [a', b']$$

It is easy to see that $H(F) = H(F_e) = H(F_r)$.

Lemma 3.3: If $F(x)$ is defined for $x \in [a,b]$ and $F(x)$ is convex or concave, then F_e can be uniquely reconstructed from $H(F)$.

Proof: From the study of convexity [Ben66, Roc70], every tangent line to F_e defines a supporting half-plane for epi_F_e. Moreover, epi_F_e can be uniquely determined by taking the intersection of all the supporting half-planes defined by the set of tangents to F_e [Ben66, Thm. 7.8]. To apply these results, we need only determine whether F is convex or concave from $H(F)$, and then determine the supporting half-planes of each tangent line.

Let the range of θ in $H(F)$ be $[\theta_s,\theta_e]$ and $\theta_e > \theta_s$. Select three tangent lines L_a, L_b, and L_c such that $\theta_s < \theta_a < \theta_b < \theta_c < \theta_e$. We first reconstruct L_a and L_c in the $x-y$ plane. These two lines will meet at some point p and divide the $x-y$ plane into four regions r_1, r_2, r_3, and r_4 (figure 3.2) such that

$$r_1 = \{(x,y) \mid x\cos\theta_a + y\sin\theta_a \le \rho_a \text{ and } x\cos\theta_c + y\sin\theta_c \le \rho_c\},$$

$$r_2 = \{(x,y) \mid x\cos\theta_a + y\sin\theta_a \ge \rho_a \text{ and } x\cos\theta_c + y\sin\theta_c \le \rho_c\},$$

$$r_3 = \{(x,y) \mid x\cos\theta_a + y\sin\theta_a \ge \rho_a \text{ and } x\cos\theta_c + y\sin\theta_c \ge \rho_c\},$$

$$r_4 = \{(x,y) \mid x\cos\theta_a + y\sin\theta_a \le \rho_a \text{ and } x\cos\theta_c + y\sin\theta_c \ge \rho_c\}.$$

If F is convex, then F must lie within r_3. If F is concave, then F must lie within r_1.

Now consider line L_b. Note that L_b cannot pass through the point p. If L_b passes through point p, then p is the only point in the intersection of r_1 and L_b (or r_3 and L_b). Hence, L_b must be tangent to F at point p. However, L_a and L_c are also valid tangent lines at point p, which violates the assumption that F is a smooth curve. Hence, L_b can only pass through r_1 or r_3 (not both). If L_b passes through r_1, then F must lie within r_1 and hence it is concave. If L_b passes through r_3, then F must lie within r_3 and is convex. □

Now we consider the implications of the second assumption on $H(C)$. First we introduce the following definitions.

Definition

A *branch* in $H(C)$ is a maximal subset of $H(C)$ such that it is

(i) a continuous curve for $0° \leq \theta \leq 180°$, and

(ii) no two points in the subset have the same θ value, and

(iii) if point q belongs to branch B_i, then q does not belong to any other branch B_j, unless q is the end point of B_i. The end point of a branch is the point that has the smallest or largest value of θ in that branch.

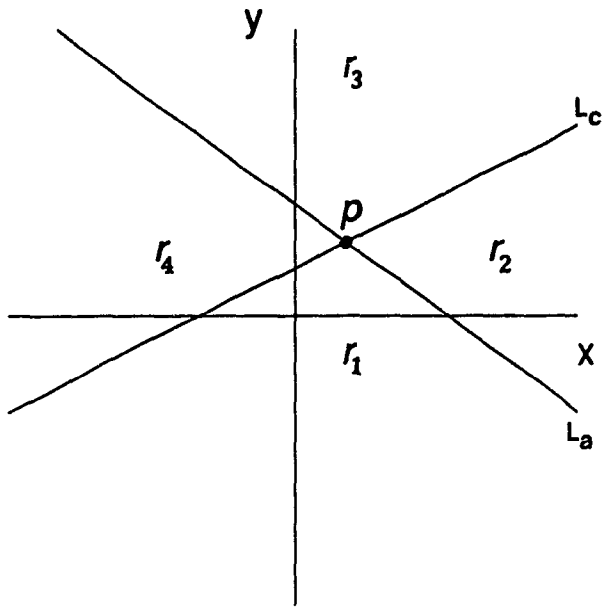


Figure 3.2 Partition of the $x-y$ plane by two non-parallel lines.

Lemma 3.4: If C satisfies assumption 2, then the following statements hold.

(i) C is formed by the concatenation of a finite number of convex/concave curve segments.

(ii) There are only a finite number of common tangent lines (lines tangent to C at more than one disjoint intervals of s).

(iii) $H(C)$ does not occupy any filled region in θ–ρ.

Proof:

(i) By lemma 3.2, a convex/concave curve segment is transformed into a continuous curve in θ–ρ with $0° \le \theta \le 180°$. For any given θ, there are only a finite number of distinct $\rho's$. Hence, $H(C)$ contains only a finite number of curves.

(ii) We will show that given two convex/concave segments F_1 and F_2, they can have at most two non-vertical common tangents.

(a) Assume F_1 is convex and F_2 is concave.

Suppose lines L_a and L_c are the two common tangents. These two lines will divide the x–y plane into four regions as in the proof of lemma 3.3. F_1 should be contained in r_3 and F_2 should be within r_1. For any other non-vertical line that is tangent to either F_1 or F_2, it can only pass through either r_1 or r_3 but not both.

(b) Assume both F_1 and F_2 are convex.

In this case, F_1 cannot touch F_2, otherwise, C is not a simple curve. Suppose lines L_a and L_c are the two common tangents. Both F_1 and F_2 have to be within region r_3. Assume $F_2 > F_1$ for all $x \in Dom\_F_1 \cap Dom\_F_2$. Let L_a be tangent to F_1 at $x = a_1$ and L_c be tangent to F_1 at $x = c_1$ (refer to figure 3.3). Since $F_2 > F_1$ for all $x \in [a_1, c_1]$, the points of tangency of L_a and L_c to F_2, namely a_2 and c_2, respectively, should satisfy $a_2 < a_1$, and $c_2 > c_1$. Now consider line L_b.

(1) $\theta_a < \theta_b < \theta_c$: L_b cannot be tangent to both F_1 and F_2. Otherwise, F_1 will be equal to F_2 at the point of tangency to F_1.

(2) $\theta_b < \theta_a$: If L_b is tangent to F_2, then it must intersect L_a to the left of a_2. Hence L_b cannot be tangent to F_1. If L_b is tangent to F_1, then it must intersect L_a in between a_1 and a_2. Hence L_b cannot be tangent to F_2.

(3) $\theta_b > \theta_c$: Similar to case (2).

(c) Assume both F_1 and F_2 are concave.

Similar to case (b).

(iii) Suppose $H(C)$ occupies a filled region $\{(\rho,\theta) \mid \rho \in [\rho_1,\rho_1+\Delta\rho]$ and $\theta \in [\theta_1,\theta_1+\Delta\theta]\}$. Then at $\theta = \theta_1+\delta$ for $0 < \delta < \Delta\theta$, there are infinite $\rho's$, contradicting assumption 2. □

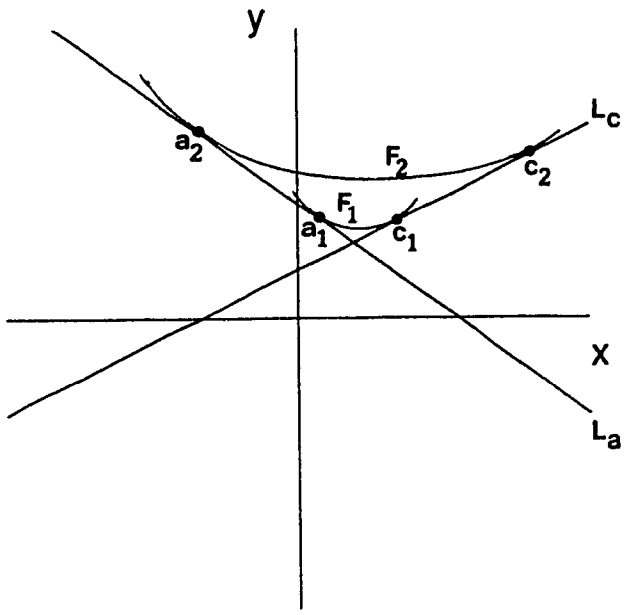


Figure 3.3 Common tangents of two convex curves.

Lemma 3.5: If the line $L_1 = (\theta_1, \rho_1)$ is tangent to $C(s)$ at k disjoint intervals $\{I_1 = [s_{11}, s_{12}], I_2 = [s_{21}, s_{22}], ..., I_k = [s_{k1}, s_{k2}]\}$, then there are $2k$ branches in $H(C)$ that meet at (θ_1, ρ_1). If $\theta_1 = 0°$ (or 180°), then the branch with end point at $(180°, -\rho_1)$ (or $(0°, -\rho_1)$) is considered to be connected to (θ_1, ρ_1).

Proof: Suppose we traverse C starting from a point which does not belong to any of the k intervals. In one complete traversal of C, the corresponding moving point $(\theta(s), \rho(s))$ in θ–ρ will visit the point (θ_1, ρ_1) k times, i.e. $(\theta(s), \rho(s))$ will move into and out of (θ_1, ρ_1) k times. Hence, there are $2k$ branches meeting at (θ_1, ρ_1). In the special case when $\theta_1 = 0°$ (or 180°), the moving point $(\theta(s), \rho(s))$ will jump to $(180°, -\rho_1)$ (or $(0°, -\rho_1)$).

Let B_i be the branch that was traced by $(\theta(s), \rho(s))$ when traversing an interval on C that is connected to I_i. The $2k$ branches have to be distinct. If these $2k$ branches are not distinct, say $B_i = B_j$, then the same segment in C is traversed more than once. This contradicts the assumption that C is a simple curve. □

Corollary 3.1: If C satisfies assumptions 1 and 2, then a line L can be tangent to C at only a finite number of disjoint intervals.

Corollary 3.2: If C satisfies assumptions 1 and 2, then $H(C)$ can always be decomposed into a unique set of finite branches $\{B_i\}$.

Proof of theorem 3.1

(i) If $C_1 = C_2$ then $H(C_1) = H(C_2)$. This part is trivial.

(ii) If $H(C_1) = H(C_2)$ then $C_1 = C_2$. To prove this, we will show that the curve C can be uniquely reconstructed from $H(C)$.

Given $H(C)$, we can always decompose it into a finite set of branches $\{B_i\}$ (by corollary 3.2). For each branch B_i, we can reconstruct the epigraph of the

convex/concave segment F_{ei} (by lemma 3.2). Hence, we can determine F_{ri} which is a subset of F_i. If point p belongs to F_i but not to F_{ri}, then p must lie on the line that is tangent to F_i at one of its end points. Let C_r' denote the union of all the F_{ri}'s reconstructed.

For a given tangent line L_i, if L_i is tangent to C at intervals $[s_{11},s_{12}]$, ..., $[s_{k1},s_{k2}]$, then the points $C(s_{11})$, $C(s_{12})$, ..., $C(s_{k1})$, and $C(s_{k2})$ are in C_r'. Moreover, for all points $p_j = C(s_j)$, for $s_j \in (s_{j1},s_{j2})$ for $j = 1$ to k, p_j lies on the straight line segment joining $C(s_{j1})$ and $C(s_{j2})$. If the original curve C is simple, then all the k straight line segments joining $C(s_{11})$ to $C(s_{12})$, ..., $C(s_{k1})$ to $C(s_{k2})$ must be disjoint and do not intersect other curve segments of C. Hence, those points in C but not in C_r' satisfy the following condition:

if $p = C(s)$ for $s \in (s_1,s_2)$ and p belongs to C but not to C_r', then p must lie on the straight line segment joining $C(s_1)$ to $C(s_2)$.

Hence, we can recover all the points in C but not in C_r' by constructing straight line segments joining the end points in C_r'.

Consider the line $L_j = (\theta_j, \rho_j)$ such that (θ_j, ρ_j) is an end point of some branches. Let $P_j = \{ p \mid p$ is an end point of some F_{ri} and L_j is tangent to F_{ri} at $p \}$. The number of points in P_j is always even (by lemma 3.5). Let $P_j = \{p_1, p_2, ..., p_{2k}\}$ and the p_i's be sorted such that $x\_p_i \le x\_p_j$ and $y\_p_i \le y\_p_j$ for $i<j$, where x_p_i is the x–coordinate of p_i and y_p_i is the y–coordinate of p_i. We then connect the pairs of points p_i to p_{i+1} for all i's that are odd. The reconstructed curve C_r is taken as the union of C_r' and all the straight line segments joining pairs of end points in C_r' as described. Note that C_r is closed, since every end point in C_r' is connected to exactly one other end point. Also, C_r is simple, since line segments connecting pairs of end points do not intersect.

By continuity, there is only one way, as described above, to join all the end points in C_r' to obtain a simple closed smooth curve. Hence, C_r must equal C.

□

3.2. Reconstruction of Open Smooth Curves

In the case of open curves, lemma 3.5 may not hold in general. However, the reconstruction procedure can still be applied with the final step modified as follows. Given a convex/concave curve F, the two semi-infinite straight lines obtained by $F_e - F_r$ are the two *semi–tangents* of F at its two end points. To connect the end points in C_r', we need to consider the direction of extension of the semi-tangents at the end points of each concave/convex segment. Let L_a be a common tangent at the two points (x_1,y_1) and (x_2,y_2) which are end points of two curves F_{ri} and F_{rj}, respectively. Assume that $x_1 < x_2$ (or $y_1 < y_2$ if L_a is vertical). Suppose the semi-tangent at (x_1,y_1) extends in the direction of positive x (or positive y if L_a is vertical) and the semi-tangent at (x_2,y_2) extends in the direction of negative x. We will then connect (x_1,y_1) and (x_2,y_2) if and only if there is no other end point of some F_{rk} that lies in between (x_1,y_1) and (x_2,y_2). The reconstructed curve C_r is identical to the original curve C, except for the two straight line segments at the two ends of C. Hence, C_r is a *reduced* version of C.

3.3. Comparisons With Horwitz's Proof and McKenzie's Work

During the course of our work, we became aware of the proof reported in [Hor89]. Horwitz's work is in a slightly different context. He only considered functions of a single variable, i.e. open curves. His proof is based on the notion of convergence. A sequence of lines $\{L_i\}$ converges to L if (i) the slope of L_i converges to the slope of L; and the y–intercept of L_i converges to the y–intercept of L. The point of

tangency of L is equal to the limit point of the intersection of $\{L_i\}$ with L. Such a limit exists provided (i) both the first and second derivatives of the function exist, and (ii) the second derivative is equal to zero at most a finite number of times. However, for common tangents[4], the limit point is not unique. Horwitz resolved this problem by simply ignoring all common tangents in his reconstruction procedure. He argued that if there are only a *finite* number of common tangents, then the finite number of points not reconstructed can be recovered by continuity constraints. The following implications can be derived from the assumptions:

(i) There are only a finite number of points of inflection on the curve.

(ii) There are only a finite number of common tangents[5].

(iii) There is no straight line segment in the curve[6].

Our proof is based on the reconstruction of the epi-graph of a convex/concave curve from its supporting half-planes. We make no assumption on the second derivative of the curve and we are able to reconstruct every point (including straight line segments) of the original curve if the curve is closed. If the curve is open, we can reconstruct the *reduced* version of the original.

Both Horwitz's method and our method are restricted to finite curves. In the case of infinite curves, for example, an infinite spiral, the trace of $(\theta(s), \rho(s))$ will eventually fill up a region in θ–ρ. Hence, we will no longer be able to decompose $H(C)$ into a set of distinct branches. Horwitz's reconstruction procedure will also fail

[4] In [Hor89], a common tangent (multiple tangent as used in [Hor89]) is defined as a line that is tangent to the curve at more than one points.

[5] This is true only for single valued functions. For closed curves or general 2D curves, such as an infinite spiral, there can be infinite number of common tangents even if the second derivative equals zero finite number of times.

[6] The difficulty in dealing with straight line segments in Horwitz's reconstruction procedure is due to the fact that the end points of the straight line segments cannot be determined.

because there are an infinite number of common tangents.

McKenzie and Protheroe described a method to reconstruct curves from the θ–ρ space [McK90]. They showed that if the curve in the θ–ρ space is differentiable (which implies that the second derivative of the curve C exist), then the point of tangency of the line to the curve C can be computed. Because of the above assumption, the same limitation of Horwitz method applies.

Chapter 4

Detection of Parametric Curves

An effective approach to reduce both the time and space requirements of the Hough transform is to decompose the high-dimensional parameterization into smaller sets of parameters that can be determined sequentially. For this method to be viable, one must formulate the problem in such a way that subsets of the parameters can be easily separated.

In this chapter, we present novel techniques to detect higher order parametric curves, such as circles and ellipses, based on the tangent representation of the curves. In the discrete domain, the transformation $H(C)$ can be computed using the modified Hough transform for straight lines (SLHT) described in chapter 2. Each detected line segment is an approximate tangent to the curve C.

First, we consider the effects of translation and rotation on the transform of a single straight line $l = (\theta_1, \rho_1)$. If the image plane is rotated by an angle[7] $\phi < 180°$ in the counterclockwise direction, then the point (θ_1, ρ_1) will be mapped to (θ_2, ρ_2) such that

$$\theta_2 = (\theta_1+\phi) \bmod 180° \quad \text{and} \quad \rho_2 = \begin{cases} \rho_1 & \text{if } \theta_1+\phi < 180°; \\ -\rho_1 & \text{if } \theta_1+\phi \geq 180°. \end{cases}$$

If the image plane is being translated by (x_0,y_0) (refer to figure 4.1), then

$$\theta_2 = \theta_1 \quad \text{and} \quad \rho_2 = t\cos(\theta_1-\alpha) + \rho_1$$

where $t = \sqrt{x_0^2+y_0^2}$, and $\alpha = \tan^{-1}(y_0/x_0)$. The transform $H(C)$ can be expressed as a

[7] If $\phi \geq 180°$, then we can perform the rotation in two steps $\phi_1, \phi_2 < 180°$ and $\phi = \phi_1 + \phi_2$.

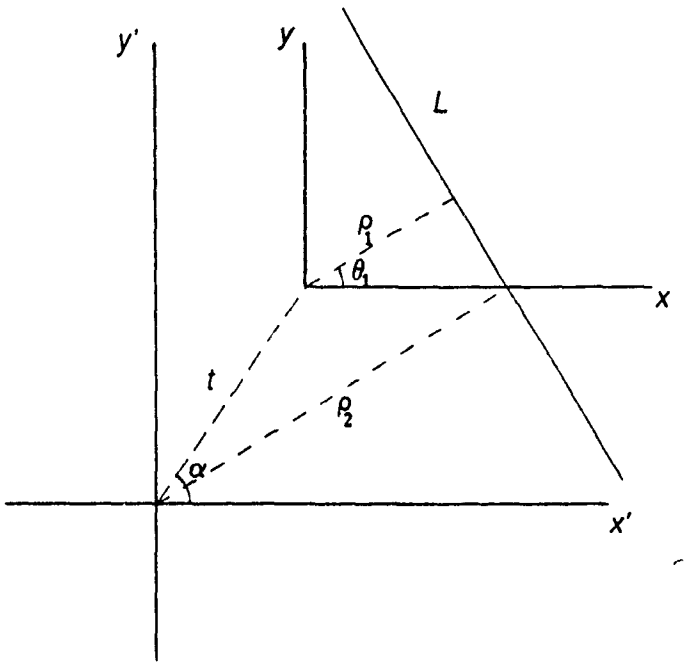


Figure 4.1 Effect of translation on the normal parameters of a straight line.

(multiple valued) function $\rho(\theta)$, where $\rho(\theta)$ is a mapping $\theta \rightarrow \rho$, such that for a given θ, say θ_i, $\rho(\theta_i) = \{\rho_j \mid (\theta_i, \rho_j) \in H(C)\}$. In the following discussion, we refer the function $\rho(\theta)$ as the SLHT transform. Let $\rho_0(\theta)$ denote the SLHT transform of the curve C located at the origin. If C' is a rotated and translated instance of C, the SLHT transform of C' is

$$\rho(\theta) = t\cos(\theta-\alpha) + sign * \rho_0(\theta-\phi)$$

where $sign = 1$ if $\theta \geq \phi$; otherwise, $sign = -1$. The "+" operator here means adding the value of $t\cos(\theta-\alpha)$ to every element in $sign * \rho_0(\theta-\phi)$. The $t\cos(\theta-\alpha)$ term is called the *translation term* and the $sign * \rho_0(\theta-\phi)$ term is called the *intrinsic term.*

We can see that this representation provides a natural decomposition of the parameter space. If we can eliminate either one of the two terms in the transform function, determining the parameters of the term left behind is relatively easy.

4.1 Detecting Circles

The SLHT transform of a circle of radius r centered at (x_0,y_0) is

$$\rho(\theta) = t\cos(\theta-\alpha) \pm r.$$

In this context, we regard the θ–ρ space as a boolean array, such that an accumulator cell will have a value 1 if it contains one or more line segments, otherwise, it will have a value 0. If the circle is properly segmented and is closed or almost closed, then we can eliminate the translation term by simply computing the difference between the pair of points that have the same θ value. Hence, we get

$$\rho'(\theta) = (t\cos(\theta-\alpha) + r) - (t\cos(\theta-\alpha) - r) = 2r$$

which is a horizontal line and its intercept with the ρ–axis is equal to two times the radius. Knowing the value of r, we can determine the center of the circle by computing an inverse transform of $t\cos(\theta-\alpha)$ [Cas87]. Each point in the θ–ρ space is mapped back into a straight line in the image plane. The set of straight lines corresponding to the sinusoidal curve $t\cos(\theta-\alpha)$ will have a common intersecting point in the image plane, which correspond to the center of the circle.

In a more general case, the circle is either not closed or it cannot be properly segmented. In this case, we try to eliminate the intrinsic term and solve for the values of t and α. Three points from the SLHT transform are selected, (θ_1, ρ_1), (θ_2, ρ_2) and (θ_3, ρ_3). To simplify the discussion we assume that the intrinsic terms of all the three ρ's selected have the same sign. The value of t and α can be obtained as follows. Let

$$\tau = \frac{\rho_1 - \rho_2}{\rho_1 - \rho_3}$$

$$= \frac{\cos(\theta_1-\alpha) - \cos(\theta_2-\alpha)}{\cos(\theta_1-\alpha) - \cos(\theta_3-\alpha)}.$$

Then we can obtain

$$\tan\alpha = \frac{(\cos\theta_1 - \cos\theta_2) - \tau\,(\cos\theta_1 - \cos\theta_3)}{\tau\,(\sin\theta_1 - \sin\theta_3) - (\sin\theta_1 - \sin\theta_2)}$$

and t can be computed by

$$t = \frac{\rho_1 - \rho_2}{\cos(\theta_1 - \alpha) - \cos(\theta_2 - \alpha)} .$$

In practice, at least two out of the three ρ's intrinsic terms will have the same sign. Suppose the sign of intrinsic term of ρ_2 is different from that of ρ_1 and ρ_3. We can then start with

$$\tau = \frac{\rho_1 + \rho_2}{\rho_1 - \rho_3} .$$

Note that there can be four different possible cases. Different combinations of set of three points are selected and the values of t and α computed. This is equivalent to performing a voting in the t–α space. If the θ axis is quantized into m_θ intervals, the points in the θ–ρ space are then divided into m_θ piles. The selection of points from the θ–ρ space follows the two guidelines below.

1. The same combination of points should not be repeated.
2. Exhaustive search over the whole domain may be prohibitive. Hence, only a limited set of combinations of piles are tried. But all possible combinations of points of each set of selected piles are considered.

One simple way to avoid repeating the same combination of piles is as follows. Suppose θ_2 is on the left of θ_1, and θ_3 is on the right of θ_1 (modulo arithmetic is used). The relationship among the three θ's is that the distance between θ_1 and θ_2 is even and that between θ_3 and θ_1 is odd. The detailed voting algorithm for circular arcs is presented below. The nesting of the for loops is understood by the indentation.

Time complexity of this process is $O(m_\theta t_s k^3)$ where k is the average number of points in each pile. The choice of w_1 and w_2 is such that the selected piles are 15° to

Voting Algorithm For Circular Arcs

```
FOR θ1 = 1 to mθ do
    select ts nonempty piles on the left of θ1 within the range w1 to w2
        whose distance from θ1 is even;
    select ts nonempty piles on the right of θ1 within the range w1 to w2
        whose distance from θ1 is odd;
    FOR i = 1 to ts do
        θ2 = the i-th selected pile on the left;
        θ3 = randomly pick one selected pile on the right;
        FOR each combination of 3 tangents, selected one from each pile do
            compute t and α;
            increment the count of the corresponding accumulator cell;
detect peaks in the t–α space;
```

45° apart from the center pile θ_1.

Because of quantization, the transform function, which is obtained by thresholding, may be noisy. Spreading of a line segment (tangent) across two adjacent accumulator cells is very common [Van81]. It is desirable to perform some filtering before voting. Using the information about the end points of the line segments, it is easy to lump connected line segments in adjacent buckets of the same column into one. The effective value of $\rho(\theta)$ of the connected line segments is taken as the weighted average, where the length of the line segment is used as the weight. Also, the boolean θ–ρ array will be sparse. Hence, we can store it in a more compact format, such that each column stores only the indexes of the cells having non-zero values.

Peaks in the t–α space are found to spread across cells in the t dimension more than in the α dimension. This is possibly due to the quantization errors associated with the values of the ρ's, and the relatively coarse resolution of α. A heuristic adaptive peak detection strategy is employed. Let N_1 be the 10 neighbors of a cell p in the same column, and N_2 be the 10 neighbors of p on adjacent columns (refer to figure 4.2). The choice of N_1 and N_2 are based on observation of the peaks in t–α of a number of cases. The 10 cells in N_2 are ranked by their votes. The collective vote of

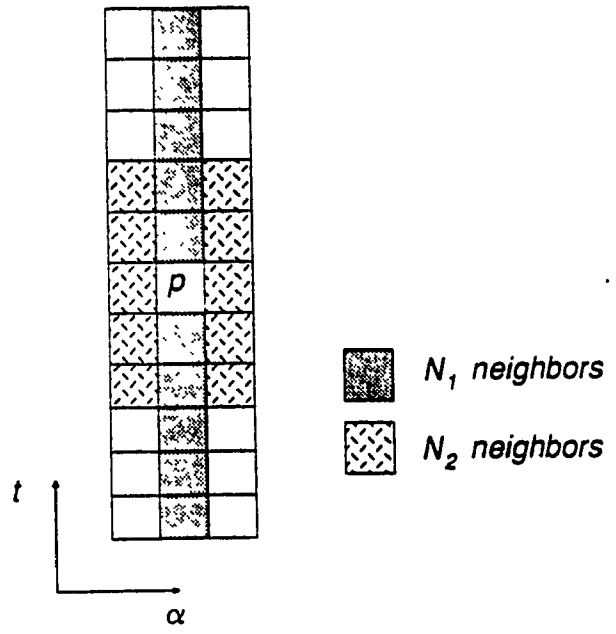


Figure 4.2 Neighbors of the cell p in t–α space.

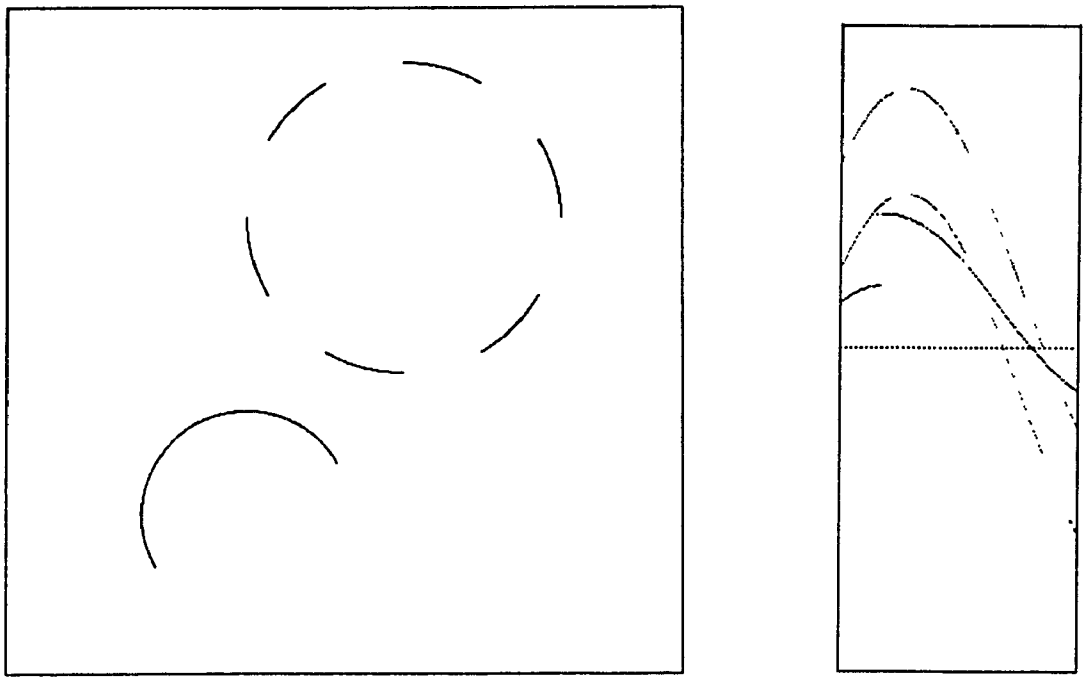
a cell p is the sum of the vote of p and its two immediate neighbors in the same column. The cell p is a peak if it satisfies all the following conditions:

1. Collective vote of p is larger than the threshold TA_{thre}.
2. Collective vote of p is larger than or equal to the collective votes of all its N_1 neighbors.
3. The collective vote of p is larger than the sum of the votes of the 3rd ranked through the 8th ranked neighbors in N_2.
4. The original vote of p is larger than or equal to the 1st ranked neighbor in N_2.
5. The original vote of p is larger than the sum of the votes of the 4th ranked through the 6th ranked neighbors in N_2.

To determine the radius of the circle, we compute a 1D histogram by first eliminating the translation term $t\cos(\theta-\alpha)$ from the transform function, and then project the boolean θ–ρ space onto the positive ρ-axis. Hypotheses of the presence of circles are

obtained only if one or more peaks in the 1D histogram can be found.

An example is shown in figure 4.3. The size of the test image is 512×512. The test image consists of a partial circle located at (180, 120) with radius equal to 80, and a dashed circle located at (300, 350) with radius equal to 120. The θ–ρ space is quantized with $\Delta\rho = 2$ units, and $\Delta\theta = 1°$. Line segments with length less than 10 units are discarded. In the voting process, T_s is set to 4 and TA_{thre} is set to 200. The expected peaks in t–α are (216.3, 33.7°) and (461, 49.4°); and the detected peaks are very close to the expected values. More examples can be found in chapter 6.



(a) Image of two partial circles; (b) The SLHT transform of (a);

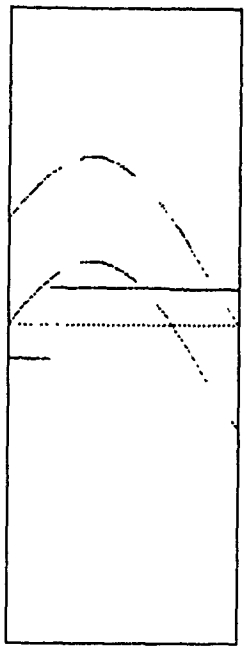
Figure 4.3 Example of circles detection.

(t) \ (α)	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
202	2	0	0	0	2	1	0	2	1	3	2	3	0	1	0
204	1	0	1	1	1	2	3	1	1	1	0	0	0	0	1
206	3	0	1	1	0	2	2	0	0	0	0	0	0	0	0
208	0	2	3	1	2	2	2	1	1	1	0	0	2	1	0
210	0	1	1	1	0	1	4	0	0	1	1	1	0	3	2
212	0	0	0	1	2	1	3	2	1	2	4	4	5	5	2
214	0	0	0	0	0	1	3	20	10	10	1	1	1	0	2
216	0	0	1	0	6	9	14	220	312	21	4	7	1	0	2
218	3	3	7	9	4	4	14	61	85	11	10	7	8	3	1
220	5	5	4	5	1	5	5	1	7	5	1	3	10	7	2
222	4	3	1	5	1	4	0	0	1	1	2	1	1	5	6
224	4	4	4	1	0	1	0	0	1	2	4	0	2	1	1
226	2	1	3	0	4	0	0	0	0	1	2	1	1	0	3
228	2	5	1	5	0	0	0	0	0	1	2	0	0	0	0
230	0	1	1	1	1	0	0	1	0	1	0	1	0	0	0

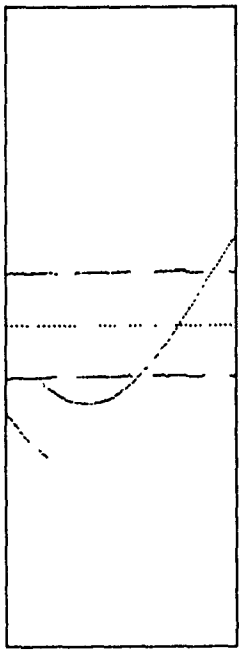
(t) \ (α)	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56
448	4	1	11	5	6	20	5	11	2	9	3	2	8	13	7
450	2	2	14	0	5	7	9	15	3	17	3	4	3	10	11
452	1	6	8	2	2	2	10	14	8	10	5	5	9	7	6
454	1	0	5	7	3	0	8	19	12	15	1	4	5	8	8
456	0	4	3	8	5	0	3	47	20	2	1	4	8	8	6
458	5	4	1	4	6	4	2	114	80	1	8	6	3	3	3
460	8	3	2	3	4	6	11	601	215	15	7	2	3	1	5
462	5	1	1	4	3	4	8	723	261	5	2	5	0	3	7
464	1	2	1	2	1	0	3	120	79	12	3	2	3	3	5
466	0	5	5	2	0	6	16	19	26	9	6	10	5	9	12
468	5	2	7	4	4	14	16	9	14	3	3	5	8	5	12
470	6	3	5	6	3	11	1	7	11	5	1	5	8	7	5
472	6	5	4	8	8	4	4	5	8	3	0	2	5	7	6
474	0	2	3	3	5	4	2	5	2	7	0	0	4	1	3
476	1	3	2	2	5	3	6	7	2	3	4	0	0	4	2

(c) Peaks detected in $t-\alpha$.

Figure 4.3 (Continued).



(d) The θ–ρ space after removing the translation term $216\cos(\theta - 34°)$.



(e) The θ–ρ space after removing the translation term $461\cos(\theta - 49°)$.

Figure 4.3 (Continued).

4.2. Detecting Ellipses

An ellipse, E, located at the origin with its major axis a and minor axis b parallel to the $x-$ and $y-$axes, respective, can be represented by

$$E: \frac{x^2}{a^2} + \frac{y^2}{b^2} = 1.$$

Let $l = (\theta_1, \rho_1)$ be the line tangent to E at (x_1, y_1), as in figure 4.4,

$$l: \rho_1 = x_1 \cos\theta_1 + y_1 \sin\theta_1.$$

The slope of line l is

$$slope_l = -\cot\theta_1 = -\frac{b^2}{a^2}\cot\psi.$$

In polar coordinates, $x_1 = r\cos\psi$ and $y_1 = r\sin\psi$. Substituting into the equation of the ellipse E and the equation of the line l, we get

$$r^2 = \frac{a^2 b^2}{b^2\cos^2\psi + a^2\sin^2\psi}$$

and

$$\rho_1 = r(\cos\psi\cos\theta_1 + \sin\psi\sin\theta_1).$$

Taking the square of the above equation, and substitute the value of r^2, we obtain

$$\rho_1^2 = \frac{a^2 b^2(\cos\theta_1 + \tan\psi\sin\theta_1)^2}{b^2 + a^2\tan^2\psi}.$$

Substituting the value of $\tan\psi$ from the equation of the slope of line l and rearranging the terms, we get

$$\rho_1^2 = a^2\cos^2\theta_1 + b^2\sin^2\theta_1.$$

Hence,

$$\rho_1 = \sqrt{a^2\cos^2\theta_1 + b^2\sin^2\theta_1}.$$

In general, the SLHT transform of an ellipse can be expressed as

$$\rho(\theta) = t\cos(\theta-\alpha) \pm sign\ \sqrt{a^2\cos^2(\theta-\phi) + b^2\sin^2(\theta-\phi)}\,.$$

We have developed two methods to detect the presence of ellipses. Both methods make use of the symmetry property of ellipses.

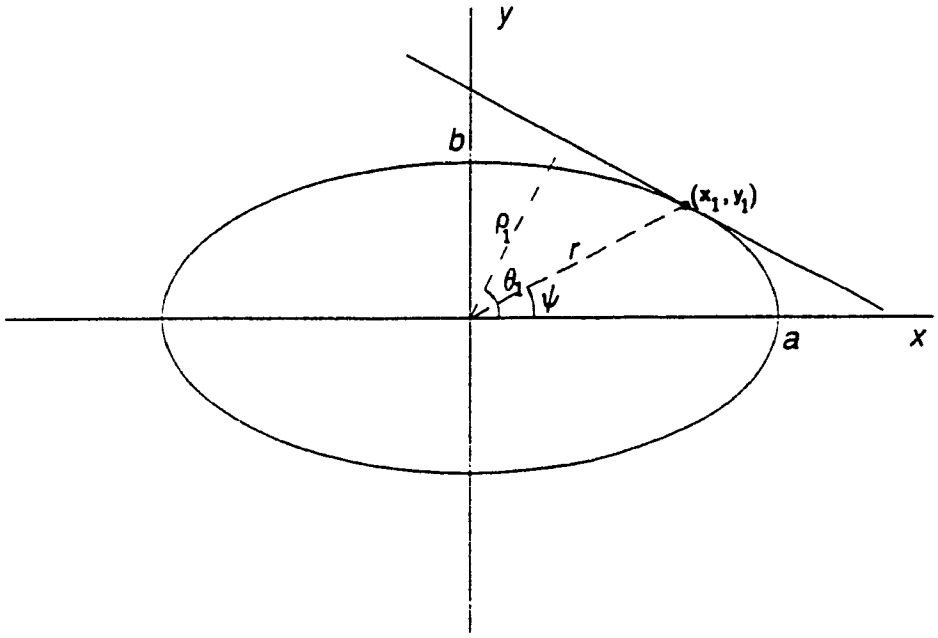


Figure 4.4 Parameterization of an ellipse.

4.2.1. Method 1

This method is essentially an extension of the method used to detect circles. If the ellipse can be properly segmented and is closed or almost closed, we can use the same trick to eliminate the translation term. This time, we will obtain a bell-shaped curve

$$\rho'(\theta) = 2\sqrt{a^2\cos^2(\theta-\phi) + b^2\sin^2(\theta-\phi)}.$$

The maximum and minimum of $\rho'(\theta)$ correspond to the values of $2a$ and $2b$,

respectively. To verify that the curve is in fact an ellipse, we can compute a 1-D correlation of $\rho'(\theta)$ with a template $T(\theta) = 2\sqrt{a^2\cos^2\theta + b^2\sin^2\theta}$. The maximum degree of correlation reveals the level of evidence and the location of the maximum indicates the orientation of the ellipse.

In other cases, we try to eliminate the intrinsic term. In general, it is not possible to solve analytically equations involving squareroot of cosine and sine functions. However, we can make use of the symmetry property of the ellipse. An ellipse is symmetrical with respect to its major and minor axes. The same holds for the intrinsic term of the SLHT transform. To solve for t and α analytically, we need to select two pairs of points from the SLHT transform, (θ_{1a}, ρ_{1a}), (θ_{1b}, ρ_{1b}), (θ_{2a}, ρ_{2a}), and (θ_{2b}, ρ_{2b}). We assume that the value of ϕ is known, and θ_{1a} and θ_{1b} are at equal distance from ϕ (or $\phi \pm 90°$) on the left and right of ϕ, respectively. Hence, the magnitude of the intrinsic terms of ρ_{1a} is equal to that of ρ_{1b}. Similarly for the other pair of points (θ_{2a}, ρ_{2a}) and (θ_{2b}, ρ_{2b}). Assume that the intrinsic term of the four points are of the same sign, then we can solve for t and α as follows. Let

$$\tau = \frac{\rho_{1a} - \rho_{1b}}{\rho_{2a} - \rho_{2b}}$$

$$= \frac{\cos(\theta_{1a}-\alpha) - \cos(\theta_{1b}-\alpha)}{\cos(\theta_{2a}-\alpha) - \cos(\theta_{2b}-\alpha)}.$$

Value of α can be determined as in the case of circular arcs. In practice, the intrinsic terms of ρ_{1a} and ρ_{1b} may be of different signs. Similarly for ρ_{2a} and ρ_{2b}. Hence, there can be four different cases to be considered. To detect an ellipse, we only need to perform an 1-D search on ϕ. Knowing t,α and ϕ, we can determine a and b by voting in the a–b space. Rewriting the SLHT transform of an ellipse,

$$a^2\cos^2(\theta-\phi) + b^2\sin^2(\theta-\phi) = (\rho(\theta) - t\cos(\theta-\alpha))^2.$$

The values of a and b can be determined by selecting two points from the SLHT

transform. The algorithm to determine a and b is similar to Algorithm1 below, except that we only need to consider a pair of tangents at a time, and the pair of tangents need not be at equal distance from the column ϕ.

Voting Algorithm1 For Elliptic Arcs

```
FOR φ = 0 to m_θ/2
   FOR t_1 = 1 to m_θ/2 −1
      θ_1a = (φ − t_1) mod m_θ;
      θ_1b = (φ + t_1) mod m_θ;
      IF either piles θ_1a or θ_1b is empty
         skip this iteration;
      FOR i = 1 to t_s
         randomly select a pair of non-empty piles θ_2a and θ_2b
            that are symmetric about φ;
         FOR each combination of 4 tangents selected on from each pile
            compute t and α;
            increment the corresponding accumulator cell;
   detect peaks in the t−α space;
determine the maximum of the peaks among all values of φ;
```

The time complexity for accumulating votes in the $t-\alpha$ space for a selected value of ϕ is $O(m_\theta t_s k^4)$, where k is the average number of points in each pile. To detect peaks in the $t-\alpha$ space requires $O(m_\theta m_\rho)$ time. Hence, the total time complexity of Algorithm1 is $O(m_\theta(m_\theta t_s k^4 + m_\theta m_\rho))$. The time complexity to determine the values of a and b is $O(m_\theta t_s k^2 + N^2)$, assuming that the accumulator array for $a-b$ is of size $O(N^2)$.

Some examples are given in section 4.2.3. One limitation of the method was observed during the simulation study. The major and minor axes of the ellipse divide the curve into four pieces, or four quadrants. If all the four points selected in the voting process correspond to lines that are tangent to E in two adjacent quadrants only, then all the terms in the expression for τ invloving α will be cancelled out. Let's

consider the case for $\phi = 0°$. Expanding the expression, we get

$$\tau = \frac{(\cos\theta_{1a} - \cos\theta_{1b})\cos\alpha + (\sin\theta_{1a} - \sin\theta_{1b})\sin\alpha}{(\cos\theta_{2a} - \cos\theta_{2b})\cos\alpha + (\sin\theta_{2a} - \sin\theta_{2b})\sin\alpha}.$$

For $\phi = 0$, we have $\theta_{1b} = 180° - \theta_{1a}$, and $\theta_{2b} = 180° - \theta_{2a}$. Hence, $\sin\theta_{1a} = \sin\theta_{1b}$ and $\sin\theta_{2a} = \sin\theta_{2b}$. All the terms involving α will be cancelled out. In the other situation, if one of the points, say (θ_{1b}, ρ_{1b}), correspond to a line tangent to E in the third quadrant, then the sign of the intrinsic terms will be different from that of ρ_{1a}. Then, we will have

$$\begin{aligned}\tau &= \frac{\rho_{1a} + \rho_{1b}}{\rho_{2a} - \rho_{2b}} \\ &= \frac{\cos(\theta_{1a}-\alpha) + \cos(\theta_{1b}-\alpha)}{\cos(\theta_{2a}-\alpha) - \cos(\theta_{2b}-\alpha)} \\ &= \frac{(\cos\theta_{1a} - \cos\theta_{1b})\cos\alpha + (\sin\theta_{1a} + \sin\theta_{1b})\sin\alpha}{(\cos\theta_{2a} - \cos\theta_{2b})\cos\alpha}.\end{aligned}$$

The values of α and t can then be computed. Hence, this method, in general, requires that segments of the ellipse be present in more than two quadrants.

4.2.2. Method 2

Instead of solving for t and α, the second method tries to locate the major and minor axes of the ellipse. The intersecting point of the major and minor axes also give the center coordinates of the ellipse.

Lemma 4.1: If $l_{1a} = (\theta_{1a}, \rho_{1a})$ and $l_{1b} = (\theta_{1b}, \rho_{1b})$ are two tangents to the ellipse E such that $\theta_{1a} = (\phi - t_1) \bmod 180°$ and $\theta_{1b} = (\phi + t_1) \bmod 180°$ where ϕ is the orientation of the major or minor axes and t_1 is a constant value, then the intersecting point of l_{1a} and l_{1b} lies on either the major or minor axes of E.

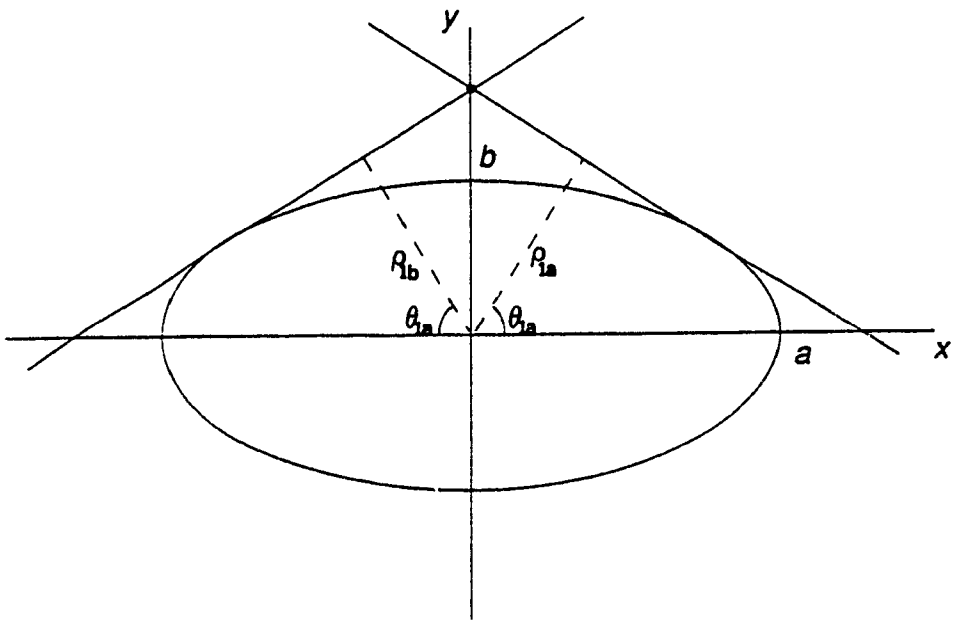


Figure 4.5 Intersecting point of two symmetric tangents to an ellipse.

Proof: Without loss of generality, we assume that the ellipse E is located at the origin and its major and minor axes are parallel to the $x-$ and $y-$axes, respectively. The two lines l_{1a} and l_{1b} are tangent to E at the first and second quadrants, respectively, as depicted in figure 4.5. We have

$$l_{1a}\colon\ \rho_{1a} = x\cos\theta_{1a} + y\sin\theta_{1a},\ \text{and}\ \ l_{1b}\colon\ \rho_{1b} = x\cos\theta_{1b} + y\sin\theta_{1b},$$

where $\theta_{1b} = 180° - \theta_{1a}$. The $x-$coordinate x_i of the intersecting point is given by

$$x_i = \frac{\rho_{1a}\sin\theta_{1b} - \rho_{1b}\sin\theta_{1a}}{\cos\theta_{1a}\sin\theta_{1b} - \cos\theta_{1b}\sin\theta_{1a}}.$$

Because of symmetry, $\rho_{1a} = \rho_{1b}$ and $\sin\theta_{1a} = \sin\theta_{1b}$. Hence, $x_i = 0$, which implies that the intersecting point lies on the $y-$axis, the minor axis of the ellipse. The same argument applies if the two lines are tangent to E at the third and fourth quadrants.

If the two lines are tangent at the first and fourth quadrants or at the second and third quadrants, then the intersecting point will lie on the major axis. Rotation and

translation of the image plane will not affect the validity of the argument. Hence the proof. □

To determine the major and minor axes, we assume the value of ϕ and select pairs of tangents. For each pair of tangents, we compute the intersecting point (x_i, y_i) and then vote for the major and minor axes.

$$\rho_a = x_i \cos(\phi + 90°) + y_i \sin(\phi + 90°),$$

$$\rho_b = x_i \cos\phi + y_i \sin\phi.$$

Two separate 1D accumulator arrays are used to accumulate votes for ρ_a and ρ_b. To increase the sharpness of the peaks, we choose to consider two pairs of tangents at the same time. The accumulator cell will be incremented only if the vote of the two pairs of tangents agree with each other. The detailed algorithm is shown below.

The time required for accumulating votes for a selected value of ϕ is $O(m_\theta t_s k^4)$, and the time for detecting peaks in the 1D histogram is $O(N)$. Hence the total time complexity of Algorithm2 is $O(m_\theta(m_\theta t_s k^4 + N))$.

Knowing the major and minor axes, we can compute the center coordinates of the ellipse. The values of a and b can be determined as in method 1. Some simulation results are given in section 4.2.3.

In the event that only half of the ellipse is present in the image, this method will only determine either the major or minor axes of the curve. In this case, we need to search along the line for the center of the ellipse and vote for the values of a and b.

4.2.3. Examples

Three examples are presented in this section. The sizes of the images are 512×512. The SLHT is computed using the same set of parameters as in the example for circles detection in section 4.1. For simplicity reason, the module used to detect

Voting Algorithm2 For Elliptic Arcs

FOR $\phi = 1$ to $m_\theta/2$
  FOR $t_1 = w_1$ to w_2 do
    $\theta_{1a} = (\phi - t_1) \bmod m_\theta$;
    $\theta_{1b} = (\phi + t_1) \bmod m_\theta$;
    IF either piles, θ_{1a} or θ_{1b} is empty
      skip this iteration;
    FOR $i = 1$ to t_s do
      randomly select a pair of non-empty piles θ_{2a} and θ_{2b} symmetric about ϕ;
      FOR each combination of 2 tangents selected from piles θ_{1a} and θ_{1b} do
        compute the intersecting point (x_1,y_1);
        FOR each combination of 2 tangents selected from piles θ_{2a} & θ_{2b} do
          compute the intersecting point (x_2,y_2);
          $\rho_{a1} = x_1\cos(\phi+90°) + y_1\sin(\phi+90°)$;
          $\rho_{a2} = x_2\cos(\phi+90°) + y_2\sin(\phi+90°)$;
          IF $|\rho_{a1} - \rho_{a2}| < \varepsilon$
            increment the entry of the 1D histogram for ρ_a;
          $\rho_{b1} = x_1\cos\phi + y_1\sin\phi$;
          $\rho_{b2} = x_2\cos\phi + y_2\sin\phi$;
          IF $|\rho_{b1} - \rho_{b2}| < \varepsilon$
            increment the entry of the 1D histogram for ρ_b;
    detect peaks in the two 1D histograms;
find the maximum among all the peaks for all values of ϕ;

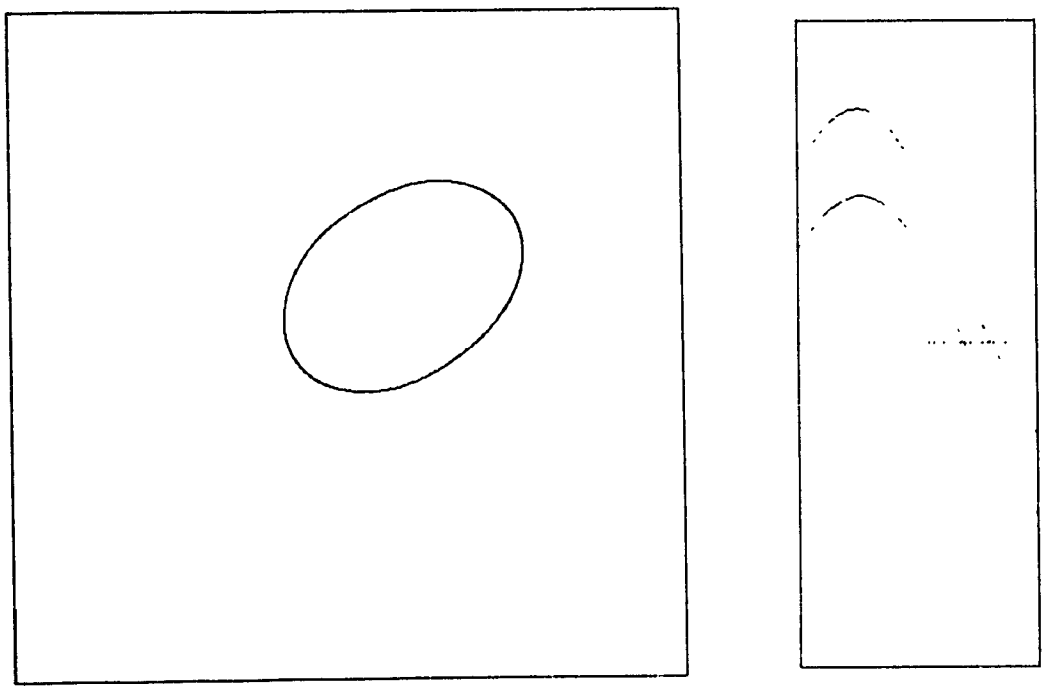
peaks in $t-\alpha$ for circles detection is used in this simulation.

Example 1

The first example consists of a single ellipse, as shown in figure 4.6a, with the following parameters: $(a, b, x_0, y_0, \phi) = (100, 70, 300, 300, 35°)$. The value of t_s in voting algorithm1 is set to 1 and the threshold value TA_{thre} is set to 150. The results of applying voting algorithm1 are summarized in figure 4.6b. The values of "votes" in figure 4.6b are equal to the sum of the accumulator cell and its two immediate neighbors in the same column. The corresponding location of the center of the ellipse in the $x-y$ plane is also shown. The detected peak in the $t-\alpha$ space for $\phi = 35°$ and its neighbors are shown in figure 4.6c. Using the detected values of t, α, and ϕ, we

perform a voting in the $a-b$ space, with t_s equal to 2, to determine the major and minor axes of the ellipse. Peaks in the $a-b$ space are detected by a simple thresholding process which takes into account the 3 by 3 neighborhood of a cell. The threshold is set to 200. The detected parameters of the ellipse are $(100, 69, 300, 300, 35°)$. The errors in the value of the minor axis is acceptable since $\Delta\rho$ is set to 2.

The results of applying the second method are summarized in figure 4.6e. In the voting process, t_s is set to 1. Again, simple thresholding is used to detect peaks in the 1D accumulator arrays for the major and minor axes. The threshold is set at 30. The expected parameters of the major and minor axes are $(73.7, 125°)$ and $(417.8, 35°)$, respectively. The detected axes are within the acceptable range.



(a) Image of an ellipse and its SLHT transform.

Figure 4.6 Example 1 of ellipses detection.

ϕ	Peaks		
	(t, α)	(x, y)	votes
20°	nil	nil	n/a
25°	nil	nil	n/a
30°	nil	nil	n/a
35°	(424, 45°)	(300, 300)	378
40°	(428, 45°)	(303, 303)	158
45°	nil	nil	n/a
50°	nil	nil	n/a

(b) Peaks detected in t–α using method 1.

	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52 (α)
410	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
412	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
414	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
416	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
418	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0
420	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0
422	0	0	0	0	0	0	0	32	0	0	0	0	0	0	0
424	0	0	0	0	0	0	0	269	0	0	0	0	0	0	0
426	0	0	0	0	0	0	0	77	0	0	0	0	0	0	0
428	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0
430	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0
432	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0
434	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
436	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0
438	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(t)

(c) Peak detected in t–α for $\phi = 35°$.

Figure 4.6 (Continued).

	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77
93	0	0	0	0	0	0	0	4	1	0	0	0	0	0	0
94	0	0	0	0	0	0	0	4	2	0	0	0	0	0	0
95	0	0	0	0	0	0	0	6	4	0	0	0	0	0	0
96	0	0	0	0	0	0	2	5	4	0	0	0	0	0	0
97	0	0	0	0	0	0	4	14	3	1	3	1	0	1	0
98	0	0	0	0	0	2	19	33	12	5	5	2	2	0	0
99	1	0	2	1	3	34	74	78	38	37	13	7	7	4	3
100	7	3	8	9	20	57	108	86	34	11	9	2	2	1	2
101	0	5	6	8	17	36	85	48	8	0	0	0	0	0	0
102	0	0	0	6	6	18	29	3	0	0	0	0	0	0	0
103	0	0	0	0	3	14	10	4	0	0	0	0	0	0	0
104	0	0	0	0	0	4	7	0	0	0	0	0	0	0	0
105	0	0	0	4	0	9	1	0	0	0	0	0	0	0	0
106	0	0	1	1	1	2	1	0	0	0	0	0	0	0	0
107	0	0	1	1	1	4	1	0	0	0	0	0	0	0	0

(a) (b)

(d) Peak detected in a–b space for $\phi = 35°$, $t = 424$, and $\alpha = 45°$.

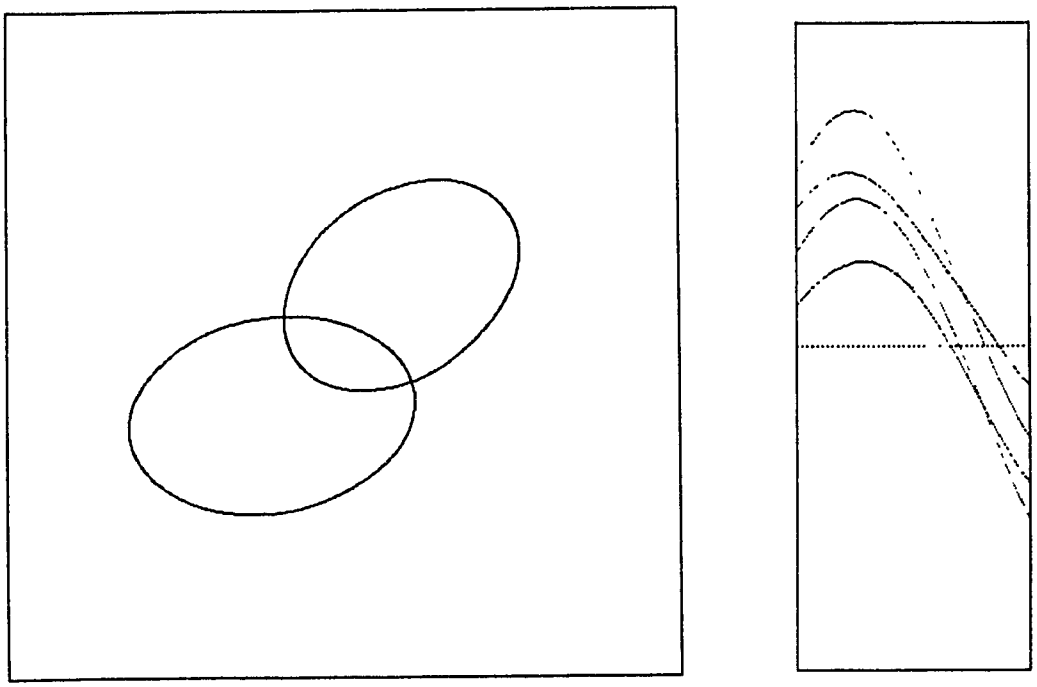
ϕ	Peaks: (ρ, votes)
20°	major axis: nil
	minor axis: nil
25°	major axis: nil
	minor axis: nil
30°	major axis: nil
	minor axis: nil
35°	major axis: (74, 217)
	minor axis: (418, 115)
40°	major axis: (38, 31)
	minor axis: nil
45°	major axis: nil
	minor axis: nil
50°	major axis: nil
	minor axis: nil

(e) Major and minor axes found using method 2.

Figure 4.6 (Continued).

Example 2

The second example consists of two overlapping ellipses with parameters equal to $(100, 70, 300, 300, 35°)$, and $(110, 75, 200, 200, 10°)$, respectively. The same voting parameter values from example 1 are used. The results of applying method 1 are summarized in figure 4.7b and 4.7c. Interferences between the two ellipses lead to the formation of secondary (spurious) peaks in $t-\alpha$. However, no peaks are found in the $a-b$ space that correspond to any of these secondary peaks. During the simulation studies, it is observed that the interferences between the two ellipses tend to reduce when they are further apart. The results of applying the second method are summarized in figure 4.7e. The expected parameters of the major and minor axes of the two ellipses are $(73.7, 125°)$ and $(417.8, 35°)$, and $(162.2, 100°)$ and $(231.7, 10°)$.



(a) Image of 2 ellipses and its SLHT transform.

Figure 4.7 Example 2 of ellipses detection.

ϕ	Peaks		
	(t, α)	(x, y)	votes
0°	nil	nil	n/a
5°	nil	nil	n/a
10°	(282, 45°)	(199, 199)	517
	(404, 34°)	(335, 226)	209
	(348, 58°)	(184, 295)	154
15°	nil	nil	n/a
20°	nil	nil	n/a
25°	nil	nil	n/a
30°	(420, 45)	(297, 297)	157
35°	(424, 45°)	(300, 300)	477
	(354, 47°)	(241, 259)	187
40°	(428, 45°)	(303, 303)	209
	(338, 46°)	(235, 243)	175
45°	nil	nil	n/a
50°	nil	nil	n/a

(b) Peaks detected in t–α using method 1.

Figure 4.7 (Continued).

	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	(α)
268	3	1	101	13	3	0	1	0	0	7	2	0	1	0	1	
270	1	7	15	13	4	2	0	0	2	11	2	0	0	3	0	
272	1	4	5	9	6	2	0	2	17	7	2	0	2	1	1	
274	1	1	4	4	12	8	0	1	15	9	0	0	1	0	2	
276	2	8	1	1	2	13	2	2	9	2	0	0	0	1	0	
278	1	6	1	1	3	10	13	2	16	4	0	1	0	2	2	
280	3	4	1	3	2	1	18	56	5	1	1	0	0	1	3	
282	2	3	1	0	2	3	4	292	14	1	0	1	2	1	2	
284	5	6	0	1	4	1	0	169	26	1	0	1	0	0	2	
286	2	1	0	2	2	0	16	21	39	4	1	0	3	2	2	
288	0	0	3	1	0	0	17	14	5	10	1	2	3	1	1	
290	3	2	0	2	0	0	23	4	1	18	6	1	0	6	3	
292	4	1	1	2	0	7	22	0	1	9	10	1	2	3	2	
294	1	0	1	1	1	10	12	2	1	2	19	9	0	2	0	
296	0	0	0	0	4	17	1	2	3	4	12	24	3	0	0	

(t)

	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	(α)
390	7	7	6	13	2	1	1	20	23	0	1	2	6	7	3	
392	2	5	9	19	3	1	4	14	16	3	1	4	5	3	3	
394	4	6	9	32	2	1	1	16	6	1	0	3	4	0	5	
396	4	3	5	17	4	0	2	26	6	2	3	1	2	2	0	
398	1	3	4	6	7	4	6	19	1	0	3	1	2	1	0	
400	3	3	4	8	3	8	2	43	1	1	5	3	1	0	1	
402	1	3	9	6	0	5	9	61	7	3	2	10	5	0	1	
404	0	0	4	7	1	3	9	91	1	1	1	4	2	4	0	
406	1	4	3	1	3	1	24	57	1	1	4	4	4	1	0	
408	2	2	5	1	2	2	30	23	8	3	2	5	2	2	3	
410	0	2	5	0	0	2	25	10	4	10	3	1	5	0	1	
412	0	2	4	0	2	0	30	4	5	7	7	3	1	2	2	
414	0	0	5	1	2	0	29	4	0	4	17	5	1	1	4	
416	1	1	2	1	0	1	10	4	2	4	13	10	2	3	5	
418	0	1	2	1	3	2	16	1	3	1	7	4	2	1	6	

(t)

	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	(α)
334	0	2	5	4	8	57	7	0	3	0	3	0	0	3	4	
336	0	7	1	5	8	25	26	4	1	2	1	2	1	3	2	
338	1	4	1	6	0	6	28	1	6	1	4	1	1	2	3	
340	2	1	4	6	7	6	44	9	2	5	1	0	2	2	2	
342	0	2	6	3	7	5	27	20	3	7	2	0	2	2	3	
344	3	0	3	6	3	1	9	35	6	2	1	0	0	1	1	
346	3	0	1	7	5	5	6	51	12	2	2	0	0	0	1	
348	1	3	3	5	3	4	7	67	30	1	0	2	0	0	0	
350	0	2	4	2	4	2	4	36	40	1	0	0	0	1	0	
352	0	2	6	3	3	4	7	9	22	3	2	0	0	2	1	
354	3	6	1	2	7	1	14	7	33	1	1	0	0	0	1	
356	3	4	1	1	0	1	3	3	12	7	1	0	1	0	0	
358	2	4	1	4	2	5	7	1	7	17	2	1	2	0	0	
360	2	2	2	3	1	10	5	2	2	12	3	0	1	0	1	
362	1	3	1	3	3	5	6	3	5	13	0	2	0	1	1	

(t)

(c1) Peaks detected in t–α for $\phi = 10°$;

Figure 4.7 (Continued).

	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	(α)
410	1	6	3	1	1	4	3	4	0	0	1	0	1	5	1	
412	1	3	0	0	0	0	2	17	1	1	0	2	0	6	2	
414	1	4	5	0	1	0	3	4	1	2	0	0	5	1	1	
416	6	12	7	4	5	2	2	5	2	0	2	0	4	5	2	
418	28	44	50	18	2	4	7	10	0	1	0	1	2	0	1	
420	19	40	84	111	32	21	7	28	2	0	0	0	4	1	3	
422	3	11	10	14	35	63	49	58	2	0	0	0	1	2	1	
424	5	7	1	2	3	10	55	317	8	2	1	0	6	0	2	
426	2	4	5	0	0	2	12	102	32	20	3	3	4	1	1	
428	0	0	4	2	2	1	2	24	5	16	22	6	7	1	0	
430	1	0	0	7	0	2	11	19	1	2	12	12	17	2	0	
432	0	5	5	1	0	0	1	9	4	0	5	18	83	15	3	
434	1	3	0	0	0	4	0	7	0	0	0	13	34	45	1	
436	0	0	0	4	2	0	1	12	0	3	3	1	5	19	31	
438	0	0	2	1	2	2	2	3	0	0	0	7	7	3	22	
(t)																

	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	(α)
340	4	3	8	3	4	4	1	13	18	3	2	1	2	0	6	
342	2	6	15	10	6	6	4	14	3	3	0	0	2	4	3	
344	2	9	5	4	8	5	2	25	3	2	2	3	0	5	10	
346	5	8	8	8	4	5	7	26	4	1	3	0	1	1	10	
348	12	23	18	11	10	5	4	32	3	3	3	1	0	4	2	
350	11	13	19	22	27	11	9	22	3	4	4	0	0	7	6	
352	0	1	13	5	5	19	11	43	4	2	4	0	1	1	2	
354	5	7	2	2	7	9	11	95	6	3	0	0	1	4	4	
356	3	2	2	1	4	0	5	49	8	5	2	3	0	6	7	
358	6	1	0	0	1	5	5	2	2	5	5	2	1	8	5	
360	2	3	0	0	0	1	2	23	0	2	5	4	1	10	5	
362	3	0	3	0	0	5	3	4	3	1	0	3	10	7	0	
364	2	3	4	3	4	1	2	20	0	2	1	1	4	17	5	
366	4	2	2	0	0	5	6	11	0	1	1	1	4	7	15	
368	3	0	0	3	1	0	7	6	1	0	1	1	3	2	9	
(t)																

(c2) Peaks detected in $\cdot - \alpha$ for $\phi = 35°$.

Figure 4.7 (Continued).

(a)	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	(b)
103	0	0	0	1	0	1	2	2	2	2	1	0	0	0	0	
04	0	0	0	0	0	0	0	4	4	2	1	0	0	0	0	
105	0	0	0	0	0	0	0	6	4	0	0	1	0	0	0	
106	0	0	0	0	0	0	8	10	7	0	0	0	0	0	2	
107	0	0	0	0	0	0	14	20	4	3	1	0	0	2	0	
108	1	0	2	0	1	15	39	46	25	3	8	5	3	4	2	
109	0	2	8	5	11	33	68	60	28	25	13	6	7	5	5	
110	1	3	4	6	18	28	61	56	21	3	0	3	2	0	1	
111	1	4	6	12	16	46	73	53	18	3	0	1	1	1	1	
112	0	5	1	10	12	14	39	10	1	0	0	1	0	0	0	
113	2	1	2	0	8	10	18	1	0	0	0	0	0	0	0	
114	2	0	1	2	2	2	4	2	0	0	0	0	0	0	0	
115	0	0	0	0	2	5	12	0	0	0	0	0	0	0	0	
116	0	0	3	0	5	6	1	0	0	0	0	0	0	0	0	
117	1	0	1	3	2	1	2	0	0	0	0	0	0	0	0	

(d1) Peak detected in a–b space for $\phi = 35°$, $t = 282$, and $\alpha = 45°$.

(a)	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	(b)
93	0	0	0	0	0	0	0	0	2	1	1	1	0	0	2	
94	0	0	0	0	0	0	0	0	5	0	1	0	3	0	1	
95	0	0	0	0	0	0	1	2	7	1	0	0	2	1	3	
96	0	0	0	0	0	0	0	2	8	3	0	0	4	0	0	
97	0	0	0	0	0	0	0	3	9	8	3	0	2	0	1	
98	0	0	0	0	0	0	1	12	38	13	7	10	2	3	0	
99	0	1	0	2	3	6	43	60	79	46	26	15	12	4	4	
100	1	9	3	5	11	17	84	109	98	34	15	9	2	2	1	
101	3	1	3	3	10	16	40	64	54	8	0	0	0	0	0	
102	0	3	1	0	3	8	30	16	3	0	0	0	0	0	0	
103	0	0	0	0	0	1	10	7	1	0	0	0	0	0	0	
104	2	0	0	0	2	2	0	9	0	0	0	0	0	0	0	
105	0	0	0	0	1	1	2	8	0	0	0	0	0	0	0	
106	0	0	0	0	0	3	4	5	0	0	0	0	0	0	0	
107	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	

(d2) Peak detected in a–b space for $\phi = 35°$, $t = 424$, and $\alpha = 45°$.

Figure 4.7 (Continued).

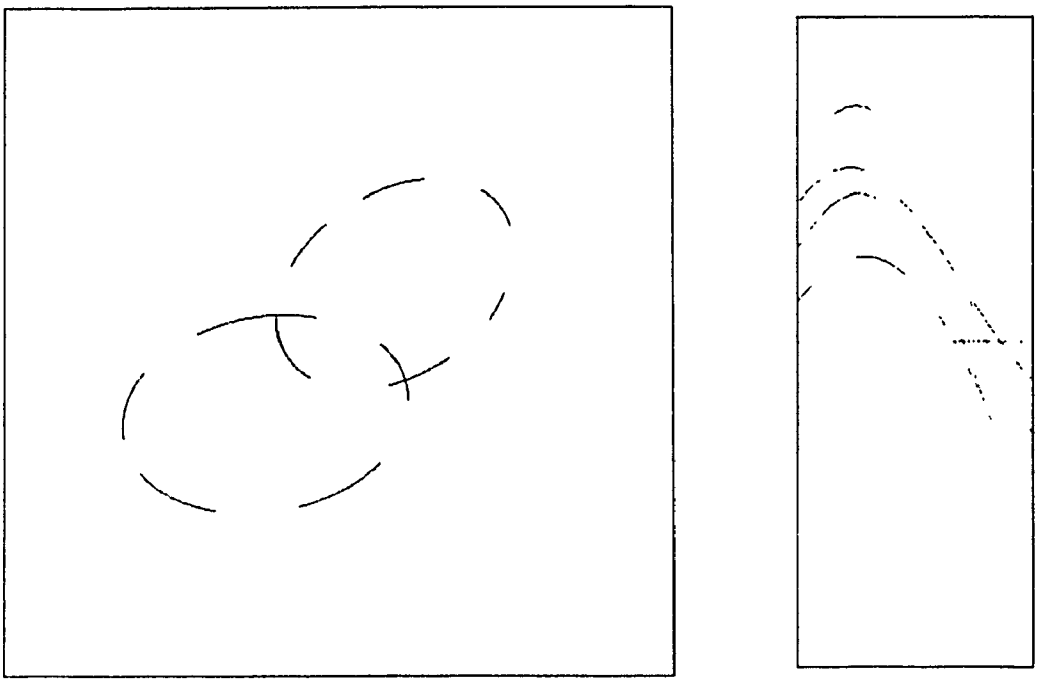
ϕ	Peaks: (ρ, votes)
0°	major axis: nil
	minor axis: nil
5°	major axis: (178, 36)
	minor axis: nil
10°	major axis: (162, 254)
	minor axis: (232, 154)
15°	major axis: (139, 42)
	minor axis: nil
20°	major axis: nil
	minor axis: nil
25°	major axis: nil
	minor axis: nil
30°	major axis: (107, 37)
	minor axis: nil
35°	major axis: (74, 232)
	minor axis: (418, 124)
40°	major axis: (36, 53)
	minor axis: nil
45°	major axis: nil
	minor axis: nil
50°	major axis: nil
	minor axis: nil

(e) Major and minor axes found using method 2.

Figure 4.7 (Continued).

Example 3

The third example consists of two overlapping dashed ellipses. The parameters of the ellipses are the same as those in example 2. The value of t_s is set to 3 and TA_{thre} is set to 150 in the voting on $t-\alpha$. In the voting on $a-b$, the value of t_s is set to 3 and the threshold is set to 150. In the voting process of method 2, t_s is set to 3 and the threshold is kept at 30. As expected, the height of the peaks are lower than that in example 2. However, the accuracy of the detected parameters are the same as that in example 2.



(a) Image of two partial ellipses and its SLHT transform.

Figure 4.8 Example 3 of ellipses detection.

ϕ	Peaks		
	(t, α)	(x, y)	votes
0°	nil	nil	n/a
5°	nil	nil	n/a
10°	(282, 45°)	(199, 199)	293
15°	nil	nil	n/a
20°	nil	nil	n/a
25°	nil	nil	n/a
30°	nil	nil	n/a
35°	(424, 45°)	(300, 300)	318
	(262, 51°)	(165, 204)	182
	(356, 47°)	(242, 259)	154
40°	(338, 46°)	(235, 243)	240
45°	nil	nil	n/a
50°	nil	nil	n/a

(b) Peaks detected in t–α using method 1.

	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	(α)
268	1	3	12	6	1	0	0	2	0	6	1	0	0	1	3	
270	2	0	0	11	6	0	1	1	2	1	0	0	0	0	1	
272	1	3	1	7	8	3	0	0	6	12	1	0	0	1	0	
274	0	3	1	2	3	15	0	0	7	8	0	0	1	0	2	
276	0	0	0	2	4	6	2	2	16	1	0	0	1	0	2	
278	2	0	2	0	2	1	9	1	6	0	0	0	1	1	1	
280	2	0	0	2	0	3	10	40	3	0	0	1	0	2	0	
282	0	0	1	1	3	1	5	179	0	0	1	0	1	0	1	
284	0	0	1	0	0	2	1	74	11	0	1	0	1	1	1	
286	0	0	0	0	0	0	17	5	5	2	2	2	1	0	1	
288	0	0	0	0	0	0	9	1	1	1	1	1	2	3	1	
290	0	0	1	0	1	1	17	0	0	8	2	3	2	5	0	
292	0	0	0	1	0	0	4	0	0	7	5	1	1	3	3	
294	0	0	0	0	1	1	1	0	1	2	14	6	4	0	3	
296	0	0	1	0	3	5	0	0	0	0	12	12	2	1	1	
(t)																

(c1) Peak detected in t–α for $\phi = 10°$.

Figure 4.8 (Continued).

	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	(α)
410	1	2	3	0	0	0	0	0	0	2	0	0	0	0	0	
412	0	0	2	0	1	1	0	0	0	0	1	0	0	0	0	
414	3	4	2	1	0	1	3	0	0	0	0	0	2	0	0	
416	0	3	7	6	1	0	2	17	0	0	0	0	3	2	0	
418	19	20	14	4	7	9	2	0	0	0	0	0	0	1	0	
420	12	16	36	64	15	5	5	43	0	0	2	1	0	0	3	
422	2	2	7	9	29	52	19	19	0	0	0	1	0	0	0	
424	2	1	0	1	2	13	30	223	1	1	4	0	6	2	0	
426	0	3	7	5	0	1	7	76	4	4	1	4	0	1	0	
428	0	0	0	0	0	4	0	9	1	13	13	2	12	0	0	
430	2	0	0	0	0	0	3	23	0	2	13	12	4	5	0	
432	0	0	0	1	0	0	0	1	0	0	5	8	51	2	1	
434	0	0	0	0	1	1	4	0	0	2	2	2	17	24	1	
436	0	0	0	0	0	0	0	3	0	0	1	1	7	12	5	
438	0	0	0	0	0	0	0	0	0	1	0	3	5	2	0	
(t)																

	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	(α)
248	0	1	0	0	4	2	0	4	9	1	0	0	0	1	0	
250	0	0	0	0	2	2	0	1	32	1	0	0	1	1	0	
252	4	6	0	0	0	2	1	8	21	0	0	2	0	0	0	
254	21	17	7	5	2	1	2	7	14	0	0	2	1	0	0	
256	4	9	8	15	8	7	2	4	9	0	0	1	1	1	0	
258	4	10	2	6	9	10	7	5	2	0	0	1	1	3	0	
260	7	15	4	12	5	0	18	54	23	0	0	0	2	0	2	
262	2	3	6	12	8	2	8	94	23	0	2	0	1	1	3	
264	4	3	1	7	8	2	9	34	1	2	2	0	0	1	0	
266	0	4	0	6	2	2	6	48	1	0	5	4	2	0	1	
268	0	0	1	0	3	3	4	32	0	2	1	7	5	3	0	
270	0	0	0	0	1	0	2	19	0	3	1	2	2	5	3	
272	0	0	0	0	0	0	0	17	0	0	3	4	7	3	5	
274	0	0	1	0	0	0	0	0	0	1	2	2	2	2	6	
276	0	0	0	1	0	0	0	0	0	0	0	0	1	3	2	
(t)																

	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	(α)
342	0	0	0	0	0	0	0	5	2	0	0	0	0	0	3	
344	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	
346	0	4	3	0	0	0	0	0	0	0	0	0	0	0	0	
348	13	13	11	1	1	1	0	0	0	0	0	0	0	0	0	
350	5	7	29	9	10	4	2	1	0	0	0	0	0	1	0	
352	3	4	5	5	14	18	5	22	0	0	0	0	0	0	0	
354	1	4	9	3	5	4	14	79	0	0	0	0	0	0	0	
356	2	2	4	6	6	4	5	47	2	0	3	0	0	0	0	
358	2	3	8	3	6	2	4	28	0	4	6	0	0	0	0	
360	0	0	1	4	9	2	9	31	1	1	6	5	0	0	0	
362	2	0	0	0	0	1	11	17	0	1	2	4	5	2	1	
364	0	3	0	0	0	0	3	28	0	1	2	3	1	11	3	
366	1	1	1	0	0	1	9	2	1	2	2	2	5	12	5	
368	0	2	0	0	0	0	2	5	0	2	1	0	3	3	9	
370	0	0	0	1	2	0	1	0	0	0	0	2	7	7	1	
(t)																

(c2) Peaks detected in t–α for $\phi = 35°$;

Figure 4.8 (Continued).

(a)	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	(b)
103	0	0	0	0	0	0	0	0	3	5	1	2	0	0	0	
104	0	0	0	0	0	0	1	0	3	1	1	0	0	0	0	
105	0	0	0	0	0	0	0	2	6	5	0	0	0	0	0	
106	0	0	0	0	0	0	0	8	10	6	0	0	0	0	0	
107	0	0	0	0	0	0	1	11	17	3	0	0	0	0	0	
108	0	0	0	0	2	5	10	31	35	15	7	4	3	1	0	
109	0	0	0	4	5	5	22	56	32	10	9	4	3	2	1	
110	2	2	10	10	10	15	40	60	46	10	8	2	0	3	2	
111	2	1	1	2	11	12	25	38	22	6	0	0	0	0	0	
112	0	0	4	4	4	7	13	9	4	0	0	0	0	0	0	
113	0	1	0	1	1	2	3	5	0	0	0	0	0	0	0	
114	0	0	0	2	0	0	2	1	1	0	0	0	0	0	0	
115	0	0	0	0	0	1	6	3	2	0	0	0	0	0	0	
116	0	0	0	0	0	2	1	0	2	0	0	0	0	0	0	
117	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	

(d1) Peak detected in a–b space for $\phi = 35°$, $t = 282$, and $\alpha = 45°$.

(a)	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	(b)
93	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	
94	0	0	0	0	0	0	0	0	4	4	0	0	0	0	1	
95	0	0	0	0	0	0	0	0	4	1	0	0	0	0	0	
96	0	0	0	0	0	0	0	1	5	6	0	0	0	1	0	
97	0	0	0	0	0	0	1	1	12	3	0	0	0	0	0	
98	0	0	0	0	0	0	0	14	19	6	0	0	1	0	0	
99	0	0	0	0	0	5	20	54	28	13	1	3	5	1	1	
100	1	2	2	5	4	14	39	66	53	15	1	2	0	0	2	
101	1	3	3	4	1	10	11	17	9	4	0	0	0	0	0	
102	0	0	0	0	0	0	7	2	2	0	0	0	0	0	0	
103	0	0	0	0	2	1	4	2	1	0	0	0	0	0	0	
104	0	0	0	0	0	3	1	5	1	0	0	0	0	0	0	
105	0	0	0	0	0	2	1	4	0	0	0	0	0	0	0	
106	0	0	0	0	0	1	1	4	0	0	0	0	0	0	0	
107	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	

(d2) Peak detected in a–b space for $\phi = 35°$, $t = 424$, and $\alpha = 45°$.

Figure 4.8 (Continued).

ϕ	Peaks: (ρ, votes)
0°	major axis: nil
	minor axis: nil
5°	major axis: nil
	minor axis: nil
10°	major axis: (162, 116)
	minor axis: (232, 81)
15°	major axis: (140, 61)
	minor axis: nil
20°	major axis: nil
	minor axis: nil
25°	major axis: nil
	minor axis: nil
30°	major axis: (108, 37)
	minor axis: nil
35°	major axis: (74, 199)
	minor axis: (418, 54)
40°	major axis: (37, 79)
	minor axis: nil
45°	major axis: (2, 32)
	minor axis: nil
50°	major axis: nil
	minor axis: nil

(e) Major and minor axes found using method 2.

Figure 4.8 (Continued).

4.3. Comparisons With Previous Works

In this section, we compare our methods, in terms of time and space complexities, with some previously published methods. To simplify the discussion, we only examine the detection of ellipses.

Let the size of the image be $N \times N$, the number of edge pixels be P, the size of the θ–ρ accumulator array be $m_\theta \times m_\rho$ and the average number of non-empty buckets per column be k. We assume that m_ρ is of order N. In the conventional Hough transform, the time required to accumulate the votes is $O(Pm_\theta N^3)$, and the peak detection cost is $O(m_\theta N^4)$. The space requirement is also $O(m_\theta N^4)$.

Ballard showed that by incorporating the gradient constraint in computing the mapping of image points to the parameter space, the number of free variables is reduced by one [Bal81]. This method is often referred to as the generalized Hough transform, GHT. Hence, the voting cost is reduced to $O(Pm_\theta N^2)$. If the estimated gradient of a pixel has an error of ε times the desired accuracy, then the voting cost may become $O(P\varepsilon m_\theta N^2)$. The peak detection cost and storage requirement remain unchanged.

Other parameter space decomposition approaches have been proposed. In [Tsuj78] and [Tsuk83], an estimate of the center coordinates is first obtained as follows. The accumulator array is the x–y plane. The accumulator cell corresponding to the mid point of a pair of edge pixels having the same gradient is incremented. The centers of the ellipses are determined by locating peaks in the accumulator array. The voting cost of this step is $O(m_\theta i^2)$, where $i = P/m_\theta$ in this case. If the error of the estimated gradient is ε times the desired accuracy, then the complexity of the process is $O(P^2\varepsilon^2/m_\theta + N^2)$, where N^2 is the peak detection cost. In [Tsuk83], the remaining three parameters of the ellipse is further divided into two sets. They are determined by a 2D transform of the whole image followed by a 1D histogramming. The

complexity of this step is $O(\upsilon(PN+N^2))$, where υ is the number of peaks found in the $x-y$ accumulator array. There are, however, two major drawbacks of this method: (i) it relies heavily on the accuracy of the gradient information; and (ii) the estimation of center coordinates may become difficult in complex images especially if parallel lines are present in the image.

Casasent and Krishnapuram [Cas87] are the first to develop object recognition methodologies using the SLHT. Assuming the orientation, major and minor axes of the ellipse, the intrinsic term of the SLHT transform can be eliminated by simple subtraction and the translation parameters can be determined by an inverse transform. The number of non-empty buckets in the $\theta-\rho$ array is $m_\theta k$. Hence, the time complexity of the inverse transform is $O(m_\theta kN+N^2)$. If the rotation and intrinsic parameters are not known in advance, then a coarse-to-fine search is carried out. To detect ellipses, it is necessary to search for the values of a, b, and ϕ. Hence, the total time complexity is approximately equal to $O(m_\theta{}^2 kN^3)$, and the space requirement is $O(N^2)$.

The time complexity of our two methods is approximately equal to $O(m_\theta{}^2(t_s k^4+N))$, and the space requirement is $O(N^2)$. The value of t_s is relatively small compared with the other parameters. In our examples, t_s is set to be less than or equal to three. The value of k can be significantly reduced in several ways:

1. Filter the $\theta-\rho$ space by lumping connected line segments that spread across adjacent buckets into one.
2. Filter out the long straight line segments (to be elaborated in chapter 5), hence the value of k is about two to three times the number of curves in the image, which is presumably a small number.

Because of the statistical nature of the voting process, we need not exhaust all the possible combinations of set of four points for each set of selected piles. We can further reduce the complexity by selecting only t_c random combinations of set of four points

instead of trying out all the k^4 possibilities. The time and space complexities of the five methods are summarized in table 4.1. The complexities of our methods are better than the other four. A more concrete comparison on the computational and functional performances of the different methods can only be obtained through a comprehensive simulation study. Because of time constraint, this is left for future research.

Table 4.1 Time and space complexities for detection of ellipses*

Methods	Time complexities	Space complexities
Conventional	$O(Pm_\theta N^3)$	$O(m_\theta N^4)$
GHT	$O(P\varepsilon m_\theta N^2 + m_\theta N^4)$	$O(m_\theta N^4)$
Casasent	$O(m_\theta{}^2 k N^3)$	$O(N^2)$
Tsukune	$O(P^2\varepsilon^2/m_\theta + \upsilon PN)$	$O(N^2)$
Pao	$O(m_\theta{}^2(t_s k^4 + N))$	$O(N^2)$

* In general, $P \gg N \approx m_\rho > m_\theta \gg k \approx \varepsilon$; t_s and υ are small numbers.

Comparing the two proposed methods, the second method is more efficient than the first one. However, the first method is more robust in detection of dashed curves and the second one tends to suffer more from sensitivity problems.

Another popular approach to speedup the computation is by iterative coarse-to-fine search [Ill87, Li86a, Mil86]. The algorithms adaptively focus on regions in the parameter space that have a high vote counts and iteratively refine the resolution of these regions. Integrating our parameter space decomposition approach with the adaptive processing techniques should result in an even more computationally efficient method.

Chapter 5
Recognition of 2D Smooth Curves

In the recognition of arbitrary shapes, we are concerned with the detection of the object as well as the estimation of the amount of translation, rotation, and scaling of the object in the image. Let $\rho_0(\theta)$ denote the SLHT transform of the reference object O located at the origin. The SLHT transform of a rotated, translated and scaled instance of O is given by

$$\rho(\theta) = t\cos(\theta-\alpha) + S * sign * \rho_0(\theta-\phi),$$

where S is the scaling factor.

An efficient technique to eliminate the translation term has been described in section 4.1 and 4.2 for detecting complete (closed) circles and ellipses. In this chapter, we will generalize the technique to recognize arbitrary shapes.

Definition

Transformation $D(\theta)$: Consider at a given θ, say θ_i, $\rho(\theta_i) = \{\rho_{i1}, ..., \rho_{ik}\}$ such that $\rho_{i1} < \rho_{i2} < \cdots < \rho_{ik}$ $(k>1)$. The set $D(\theta_i) = \{\rho_{il} \mid \rho_{il} = \rho_{ij} - \rho_{i1}$ for $1<j\leq k$; or $\rho_{il} = \rho_{ik} - \rho_{ij}$ for $1<j<k\}$. Hence, the number of elements in $D(\theta_i)$ is equal to $2k-3$.

In transformation $D(\theta)$, we have effectively computed the perpendicular distances between parallel tangent lines.

Lemma 5.1: The $D(\theta)$ transform of an object O possesses the following properties.

(i) It is invariant to translation of O in the image plane.

(ii) If O is rotated counterclockwise by an angle ϕ, then $D(\theta) = D_0(\theta-\phi)$.

(iii) If O is scaled by a factor S, then $D(\theta) = S * D_0(\theta)$.

Proof of lemma 5.1 is straightforward and is omitted here. Given a reference template, $D_0(\theta)$, and the transform of the test pattern $D(\theta)$, to determine the orientation of the test pattern, we need only to perform a 1D correlation of the test template with the normalized reference template. This will be elaborated in section 5.1. The peak of the correlation indicates both the likelihood of existence and the orientation of the object.

In general, the $D(\theta)$ transform is not unique. Consider smooth convex objects of constant width [Cha74]. These objects will all have their $D(\theta)$ transform equal to a horizontal straight line. Hence, the $D(\theta)$ transform can only be used as a *signature* of the object. We call this signature the STIRS (*Scalable Translation Invariant Rotation-to-Shifting*) signature. The procedure to recognize an object is as follows:

step 1: Compute the transforms $\rho(\theta)$ and $D(\theta)$ of the test pattern.

step 2: Normalize the reference template D_0 with respect to the test template D.

step 3: Determine the orientation of the object by computing a 1D correlation of the reference template with the test template. If no significant peak is detected, then we can conclude that the test pattern is not an instance of the reference object. The position of the peak indicates the amount of rotation ϕ (or ϕ+180° since $D(\theta)$ is periodic in 180°) of the test pattern.

step 4: Eliminate the intrinsic term in $\rho(\theta)$ by "subtracting" the scaled and shifted reference template $S * sign * \rho_0(\theta-\phi)$. Since $\rho(\theta)$ is a multiple valued function, the subtraction function is basically a correlation function between the two "vectors" $\rho_0(\theta_i-\phi)$ and $\rho(\theta_i)$ for each θ_i. This will be elaborated in section 5.1. The value of t and α can then be determined by an inverse

transform to the $x-y$ plane or by performing a voting in the $t-\alpha$ space. The peak corresponds to the location of the test pattern in the image plane. If no peak is found, then the test pattern is not an instance of the reference object.

We can further speed up the classification of the test pattern by computing a third transform by projecting the $D(\theta)$ transform onto the ρ–axis.

Definition

The *profile* of the projection of $D(\theta)$ onto the ρ–axis, $\Gamma(\rho)$, is equal to the number of points in $D(\theta)$ that have the same ρ.

Obviously, the transform $\Gamma(\rho)$ is invariant to both translation and rotation. Hence, step 2 above can be modified such that if the profile of the test pattern does not match that of the reference object, then we can conclude that the test pattern is not an instance of the reference object.

5.1. Implementation Details

The $D(\theta)$ transform is represented by a 2D array $D[m_\theta][m_\rho]$ such that

$$D[\theta_j][\rho_i] = \begin{cases} 1 & \text{if } \rho_i \in D(\theta_j), \\ 0 & \text{otherwise} \end{cases}$$

To find out the rotation of the test pattern, we simply slide the test template (with wrap around) over the reference template. The position at which maximum correlation occurs corresponds to the amount of rotation ϕ (or ϕ+180°). Let A_0 and D_0 be the $\rho(\theta)$ and $D(\theta)$ transforms of the reference object, respectively. Similarly A_t and D_t are the transforms of the test pattern. The detailed algorithm is listed below. The nesting of the for loops and the if-else statements are understood by the indentation.

The time complexity of this process is $O(m_\theta{}^2 m_\rho)$. However, we can easily improve the efficiency by adopting compact storage for D_r and D_t. The time

Matching Algorithm

```
FOR θ1 = 0 to mθ − 1
   FOR ρ1 = 2 to mρ/2
      IF (Dt[θ1][ρ1] = 1)
         FOR θ2 = 0 to mθ − 1
            t = (θ2 − θ1) mod mθ;
            IF (Dr[θ2][ρ1]) = 1)
               score[t] = score[t] + 2;
            ELSE IF (Dr[θ2][ρ1±1] =1)
                  score[t] = score[t] + 1;
               ELSE
                  score[t] = score[t] − 1;

   FOR θ2 = 0 to mθ −1
      t = (θ2 − θ1) mod mθ;
      FOR ρ1 = 2 to mρ/2
         IF (Dr[θ2][ρ1]) = 1 and Dt[θ1][ρ1] = 0 and Dt[θ1][ρ1±1] = 0)
            score[t] = score[t] − 1;

/* normalize the score;
   count = average number of points in D0 and Dt  */
FOR θ1 = 0 to mθ − 1
   score[θ1] = score[θ1] × 100 / count;
detect peak in score[];
```

complexity may then become $O(m_\theta^2 K_D)$, where K_D is the average number of points in each column vector of D_r and D_t. Once we know S and ϕ, we can determine t and α by "subtracting" $S * sign * \rho_0(\theta)$ from $\rho_t(\theta)$. Recall that the effect of translation is the same as adding the offset $t\cos(\theta_i - \alpha)$ to each element of $\rho_0(\theta_i)$. Determining the value of the offset is equivalent to finding the position at which maximum correlation of the two vectors $A_0[\theta_i]$ and $A_t[\theta_i]$ occurs. The correlation function is similar to that described in the matching algorithm. The only difference is that we now have 1D vectors instead of 2D templates. Let $T[m_\theta]$ denote the resultant translation term obtained for each θ_i. In case there is no correlation between the two vectors, the corresponding entry in $T[m_\theta]$ will be set to *nil*. To determine the value of t and α, we can perform

either an inverse transform to the $x-y$ plane or a voting in the $t-\alpha$ space as in the case of detection of circles.

5.1.1. Quantization errors

There are two sources of quantization errors, the digitization of the image plane and the quantization of the $\theta-\rho$ space. In digital images, the notion of "smoothness" is not well defined. The resolution of the digitization will determine how well we can estimate the gradient of the curve. This problem is further complicated by the quantization of the $\theta-\rho$ space. Because of discretization, spreading of long straight line segments across adjacent accumulator cells is very common [Van81]. In order to capture the tangents at places with small radius of curvature, a lower threshold should be used. However, lowering the threshold will certainly increase the noise (spurious tangents) in the transform space due to spreading of straight line segments. Two suggestions to solving this problem are proposed.

1. To reduce the noise in the $\theta-\rho$ space due to spreading of straight line segments across two or three adjacent accumulator cells, we can lump connected line segments in adjacent buckets of the same column (same θ) into one. This can be done since the information about the end points of the line segments are available. The effective value of ρ is taken as the weighted average, where the length of the line segment is used as the weight.
2. To remove spurious tangents, we can first extract long straight lines from the $\theta-\rho$ space. The selected straight lines are then reconstructed. For any line segment in $\theta-\rho$ whose length is less than some threshold L_t, if its constituent pixels are included in the reconstructed image then the line segment will be removed.

These two noise filtering strategies will be illustrated in the examples in section 5.1.2.

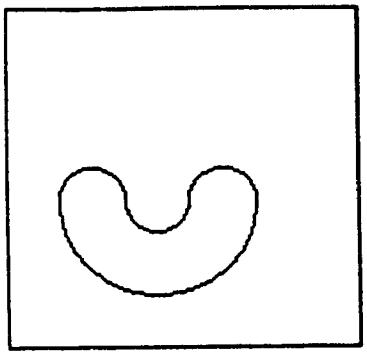
5.1.2. Examples

The size of the images used in the examples are 128×128. The θ–ρ space is quantized with $\Delta\rho = 2.0$ units, and $\Delta\theta = 2°$, line segments of length less than 5.0 units are discarded (except for example 3).

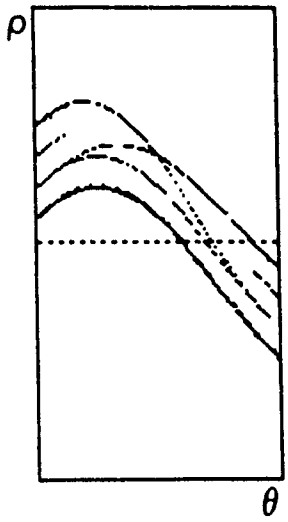
Example 1

Figure 5.1(a1) shows a test object O_1, which is a scaled instance of the shape shown in figure 3.1. The shape is composed of four half-circles. The center of mass (CG) of the shape is chosen as the reference point. The transforms of the reference shape are generated by program. We can easily observe the similarity of the profiles, the $\Gamma(\rho)$ transforms, of the test object and the reference. The result of matching the $D(\theta)$ transform of O_1 and the normalized transform of the reference object is shown in figure 5.1(c). The normalization was done in the discrete domain. The orientation of O_1 was correctly determined. Figure 5.1(d) depicts the translation term of $\rho_1(\theta)$ after eliminating the intrinsic term, and the result of the voting process is highlighted in figure 5.1(e). The estimated translation of O_1 was offset by 2 pixels. This offset was acceptable since we have $\Delta\rho = 2$.

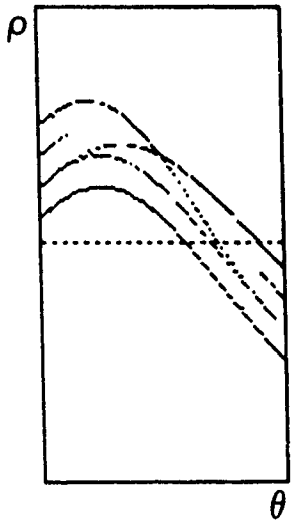
In figure 5.1(f1), we introduce 4% random noise into the image (i.e. 4% of the image is random noise). We compute the transform of this noisy image and match it with the original (without noise). There are only about 11 points out of 337 points in the $D(\theta)$ transform that do not match.



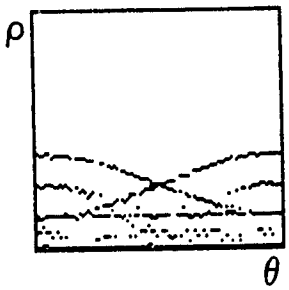
(a1) a scaled instance of the shape shown in fig. 3.1;



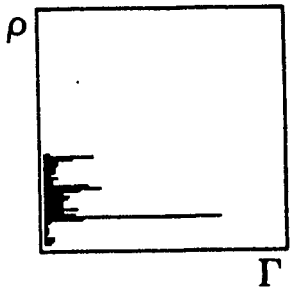
(a2) ρ(θ) transform of (a1);



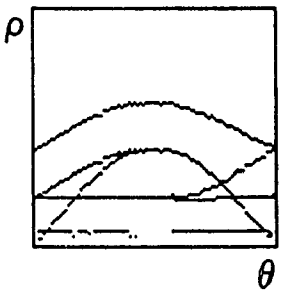
(a3) the filtered ρ(θ);



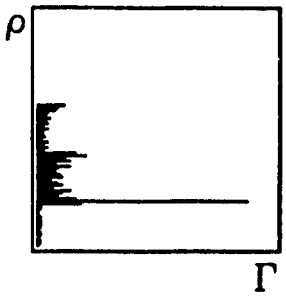
(a4) D (θ) transform of (a3);



(a5) profile of (a4);

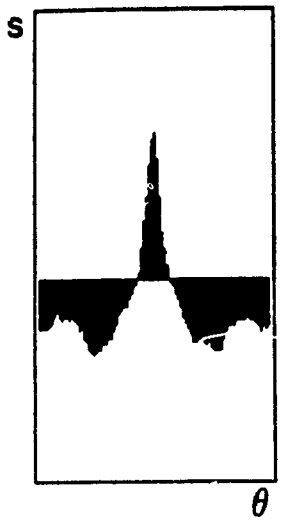


(b1) D (θ) transform of the shape shown in fig. 3.1;

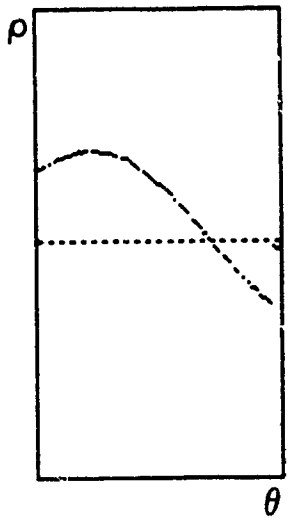


(b2) profile of (b1);

Figure 5.1 Example 1 of object recognition using the STIRS signature.



(c) result of matching (a4) against the normalized template (b1), peak detected at $(\theta, score) = (90°, 109)$;

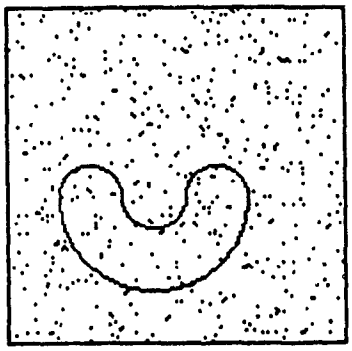


(d) the translation term of (a3) obtained by eliminating the intrinsic term;

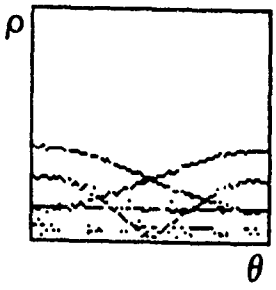
(t) \ (α)	33	34	35	36	37	38	39	40	41	42	43	44	45
65	1	0	1	0	0	0	0	0	0	0	0	0	0
66	1	1	2	5	1	5	0	0	0	0	0	0	0
67	0	1	4	3	3	3	0	0	0	0	0	0	0
68	1	6	8	14	20	16	0	1	0	0	0	0	0
69	0	2	5	16	49	40	15	1	0	0	0	0	0
70	1	3	6	6	23	56	56	21	5	1	3	0	0
71	0	0	3	0	5	24	61	15	5	1	1	0	0
72	0	0	0	2	0	12	22	18	10	2	0	0	0
73	0	0	0	0	0	4	10	12	9	3	1	0	0
74	0	0	0	0	0	0	8	6	5	1	1	2	0
75	0	0	0	0	0	0	2	4	10	1	1	2	0
76	0	0	0	0	0	0	0	1	2	2	2	0	0
77	0	0	0	0	0	0	0	1	2	3	1	1	1

(e) result of voting in $t-\alpha$ space, expected peak is at (69.8, 38°), and the detected peak is at (70.5, 38.5°);

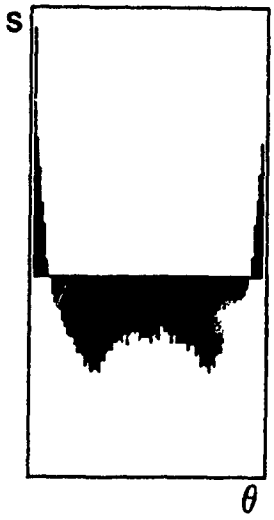
Figure 5.1 (Continued).



(f1) image of (a1) corrupted with 4% random noise;



(f2) $D(\theta)$ transform of (f1);



(f3) result of matching (f2) against (a4), peak detected at $(\theta, score) = (0°, 184)$;

Figure 5.1 (Continued).

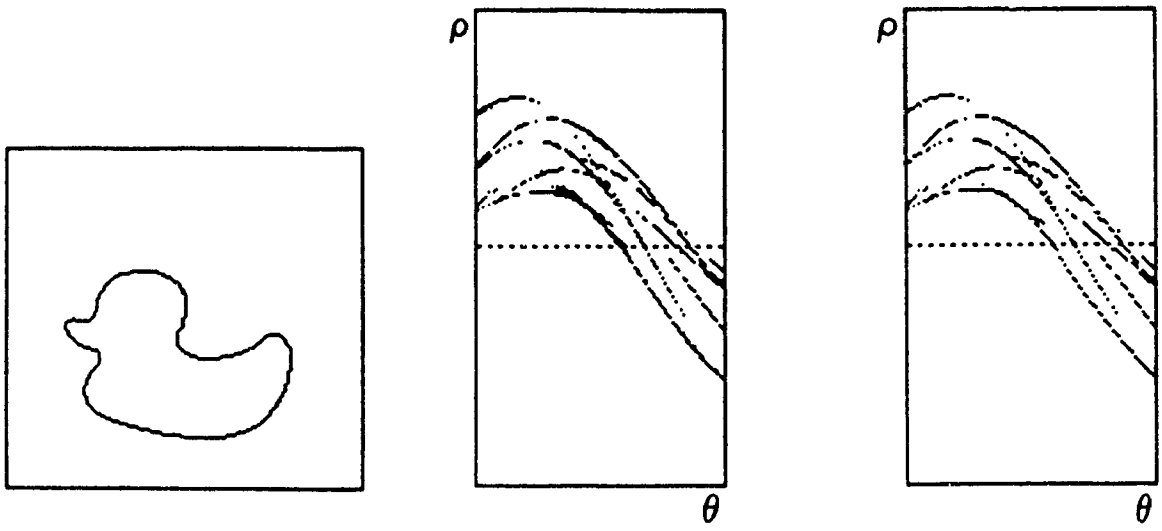
Example 2

Figure 5.2(a1) shows another reference object O_{2a} which is the outline of a duck drawn by free hand. The test object shown in figure 5.2(b1) is obtained by transposing the reference object. The CG of the reference is at (59,45). The intrinsic term $\rho_0(\theta)$ of the reference object is obtained by

$$\rho_0(\theta) = \rho_{2a}(\theta) - 74.2\cos(\theta - 37.3°).$$

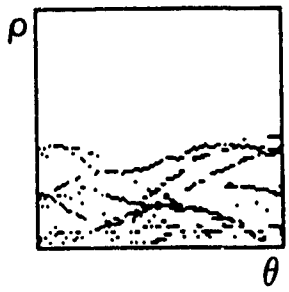
The orientation of the test object was correctly determined by matching the $D(\theta)$ transforms. The estimated location of the test object was offset by 3 pixels in this example.

We also try to match the $D(\theta)$ transform of this shape with the reference used in example 1. In figure 5.2(f), we observe no correlation between the two templates.

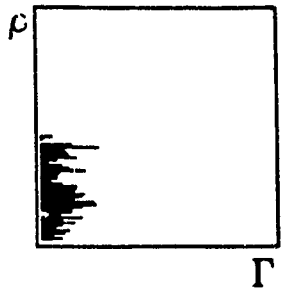


(a1) image of object O_{2a}; (a2) $\rho(\theta)$ transform of (a1); (a3) the filtered $\rho(\theta)$;

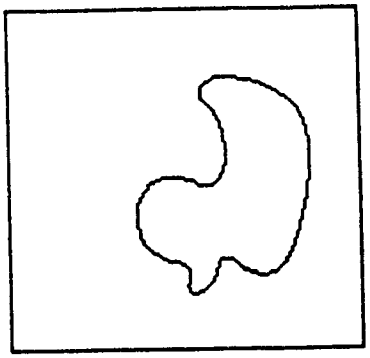
Figure 5.2 Example 2 of object recognition using the STIRS signature.



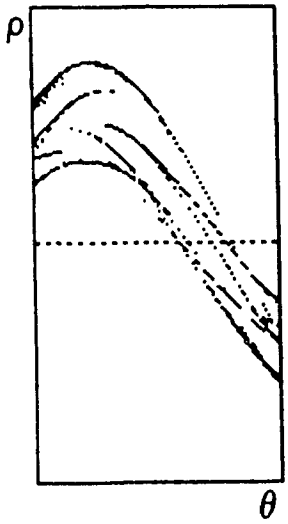
(a4) $D(\theta)$ transform of (a3);



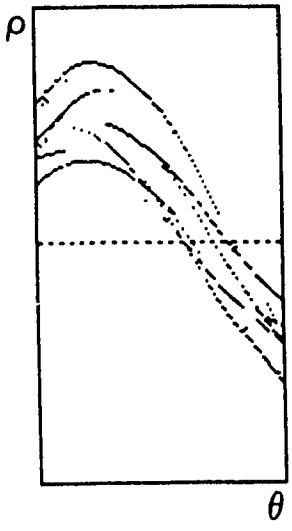
(a5) profile of (a4);



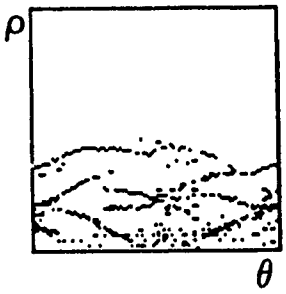
(b1) object O_{2b} obtained by transposing (a1);



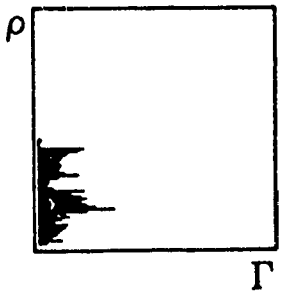
(b2) ρ(θ) transform of (b1);



(b3) the filtered ρ(θ);

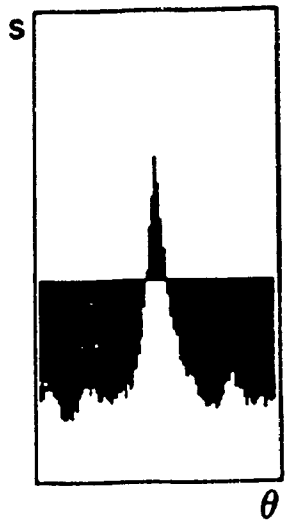


(b4) $D(\theta)$ transform of (b3);

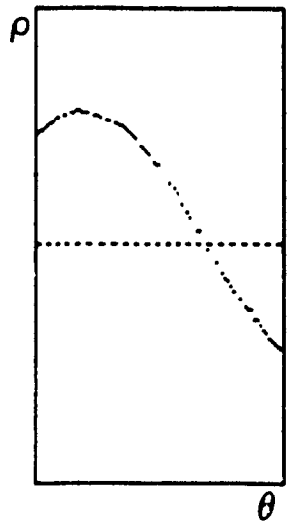


(b5) profile of (b4);

Figure 5.2 (Continued).



(c) result of matching (b4) against (a4) peak detected at $(\theta, score) = (90°, 90)$;

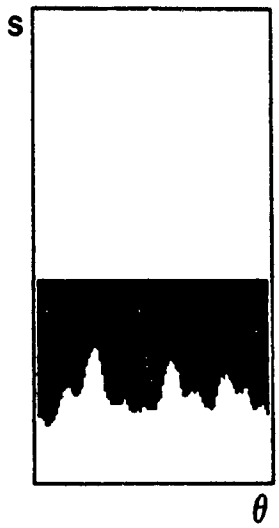


(d) the translation term of (b3) obtained by eliminating the intrinsic term;

(t) \ (α)	28	29	30	31	32	33	34	35	36	37	38	39	40
92	0	0	0	0	0	1	0	0	0	0	0	0	0
93	0	0	0	0	2	2	3	2	0	0	0	0	0
94	0	0	0	0	1	2	2	2	0	0	0	0	0
95	0	1	0	1	1	5	3	2	0	0	0	0	0
96	1	0	0	1	2	16	12	2	0	0	0	0	0
97	0	1	0	2	2	7	17	2	0	0	0	0	0
98	0	0	0	3	1	7	39	17	5	1	0	0	0
99	0	0	0	0	4	5	24	22	4	3	3	2	0
100	1	0	0	0	1	7	19	39	18	7	1	5	1
101	1	1	0	0	0	1	3	15	15	15	10	3	2
102	0	0	0	0	0	0	1	4	4	12	3	8	2
103	0	0	0	0	0	0	0	0	5	5	5	3	0
104	0	0	0	0	1	0	1	0	1	2	1	0	3

(e) result of voting in t–α space, expected peak is at (101, 35.7°), and the detected peak is at (99, 34.5°);

Figure 5.2 (Continued).

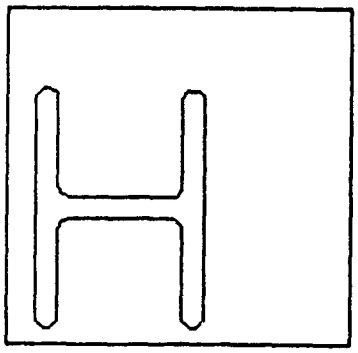


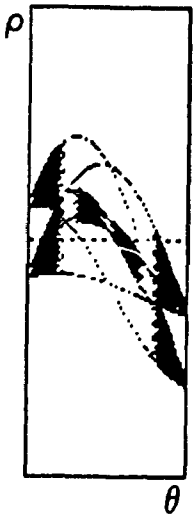
(f) result of matching (a4) against the normalized template of figure 5.1(b1), no correlation is observed.

Figure 5.2 (Continued).

Example 3

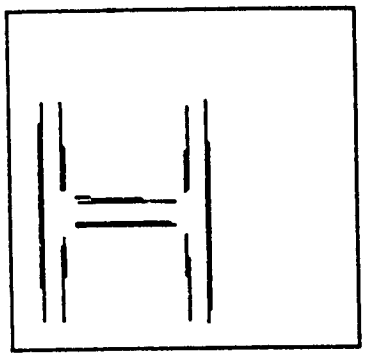
The pattern used in this example, figure 5.3(a1), is the outer contour of the character H. This pattern has long straight lines and sharp turns. We also try to compute the $\rho(\theta)$ transform with a lower angular resolution with $\Delta\theta = 3°$. In order to capture the tangents at the corners, a lower threshold of 3.0 units is used. Line segments whose lengths are greater than 20 units are selected. The reconstruction of the selected long straight line segments are shown in figure 5.3(a3). In the reconstruction, the selected line segments are first shortened by 2 units at both ends so that we will not remove the short tangents near the corners. The $\rho(\theta)$ transform before and after filtering are shown in figure 5.3(a2) and 5.3(a4), respectively. Figure 5.3(b1) shows an instance of the same pattern rotated by 24° in the clockwise direction, and figure 5.3(c) shows the result of matching the $D(\theta)$ transforms of the test pattern with the reference pattern. Almost the same correlation score is obtained if we use a higher angular resolution with $\Delta\theta = 2°$ (refer to figure 5.3(d)).



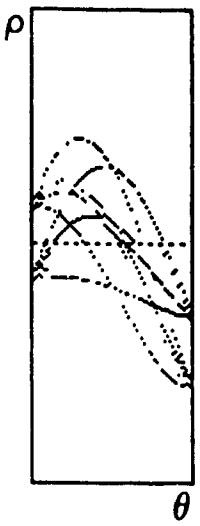


(a1) outer contour of the letter "H"; (a2) $\rho(\theta)$ transform of (a1);

Figure 5.3 Example 3 of object recognition using the STIRS signature.

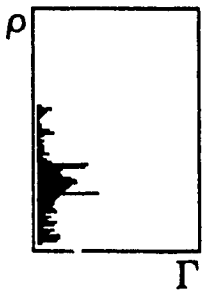


(a3) reconstructed image from the long straight segments detected in (a2);



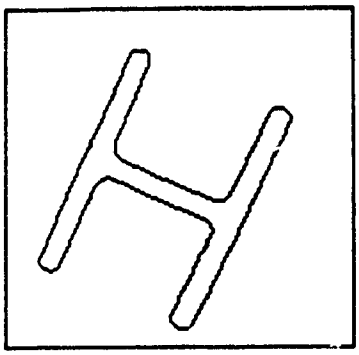
(a4) the filtered ρ(θ);

(a5) D (θ) transform of (a4);

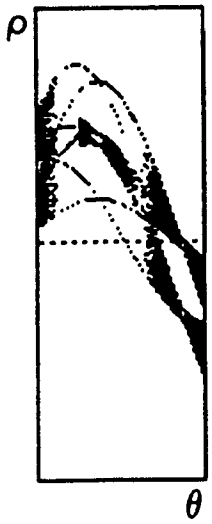


(a6) profile of (a5);

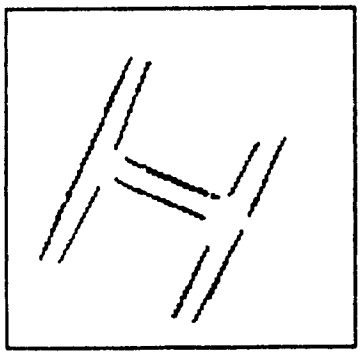
Figure 5.3 (Continued).



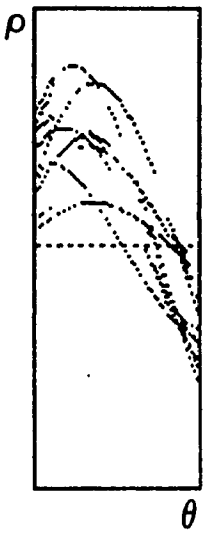
(b1) outer coutour of the letter "H" rotated by 24°;



(b2) ρ(θ) transform of (b1);



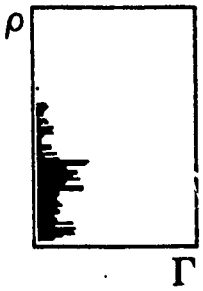
(b3) reconstructed image from the long straight segments detected in (b2);



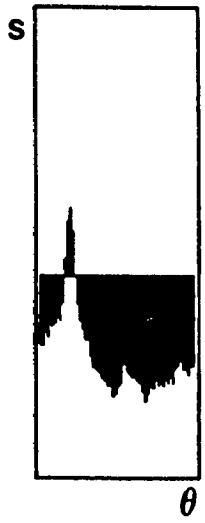
(b4) the filtered ρ(θ);

Figure 5.3 (Continued).

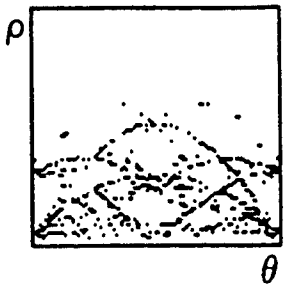
(b5) $D(\theta)$ transform of (b4);



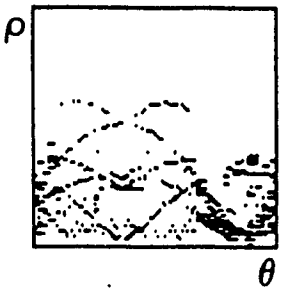
(b6) profile of (b5);



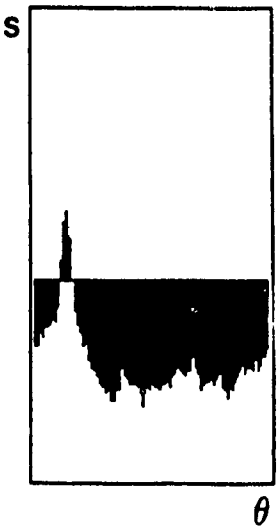
(c) result of matching (b5) against (a5);



(d1) $D(\theta)$ transform of (a1) with higher angular resolution;



(d2) $D(\theta)$ transform of (b1) with higher angular resolution;



(d3) result of matching (d2) against (d1).

Figure 5.3 (Continued).

In the above three examples, we can observe that the profiles ($\Gamma(\rho)$ transforms) of the $D(\theta)$ transforms, although quite noisy, can be used to distinguish the three different shapes.

5.2. Performance of the Recognition Method

In this section, we analyze the computational efficiency and the sensitivity aspects of the matching method presented in the previous section. The computational complexity of our method is analyzed and compared with the standard template matching and the generalized Hough transform (GHT) [Bal81]. We have seen examples of matching the transforms of two identical shapes in section 5.1. In section 5.2.2, we will give a qualitative discussion on the sensitivity of the $D(\theta)$ transform to shape distortion.

5.2.1. Computational Complexity

Assume that the input image is of size $N \times N$ and the number of black pixels is P (the length of the object contour). The computational complexity of our method can be divided into two parts, namely the overhead and the matching cost. The overhead includes computing the transforms, and filtering the θ–ρ space. To simplify the comparison, we assume that the algorithms are implemented in a sequential machine. To implement the SLHT transform described in chapter 2, we need to sort the pixels. This can be done by simply reading out the pixels in row major or column major order. Hence, the complexity to compute the SLHT is $O(N^2+Pm_\theta)$. The time to compute the $D(\theta)$ transform is $O(m_\theta k)$ where k is the average number of points in each column of $\rho(\theta)$. The computation cost to filter the transform space is $O(N^2+m_\rho m_\theta)$. Hence, the total overhead is $O(N^2+Pm_\theta+m_\theta k+m_\rho m_\theta)$, which is approximately equal to $O(N^2+Pm_\theta)$.

The algorithm to match $D(\theta)$ against the reference template takes $O(m_\theta{}^2 K_D)$ time where K_D is the average number of points in each column of the $D(\theta)$ transform,

and the time required to compute the translation term is $O(m_\theta k)$. And finally, the voting algorithm to determine t and α has a complexity of $O(m_\theta{}^2)$. Suppose there are η different object classes. To determine the class of a test pattern, we only need to match the $D(\theta)$ transform of the test pattern with the η reference templates. Assume that only one reference template matches with the test pattern, then the total matching cost is $O(\eta m_\theta{}^2 K_D + m_\theta K_D + m_\theta{}^2)$ which is approximately equal to $O(\eta m_\theta{}^2 K_D)$. The complexity of the matching process can be further brought down to $O(\eta m_\rho + m_\theta{}^2 K_D)$ if the $\Gamma(\rho)$ transform of the objects are used to reject incorrect hypothesis.

Now we consider the standard template matching. There is no overhead involved in this method. Let the template size be $T \times T$, and the search range of the allowable translation be R. Then the matching cost of this method is $O(\eta m_\theta R^2 T^2)$, where $R^2 T^2$ is at least $O(N^2)$.

In the GHT [Bal81], the gradients of each contour pixels are first computed. Let κ be the average number of pixels having the same gradient, i.e $\kappa = P/m_\theta$. Then the matching cost of GHT is $O(\eta(m_\theta P \kappa + \textit{cost of peak detection in image plane}))$. Assume that the votes are mapped to a region in the image plane of size $n \times n$. Then the complexity is equal to $O(\eta(P^2 + n^2))$.

Comparing the three methods, our method involves a feature extraction phase (computing the transforms). Because of the invariant properties of the transforms, the matching cost of our method is much lower than the other two methods. The advantage of our method is even more prominent when the number of object classes is large, i.e η is large.

5.2.2. Sensitivity To Shape Distortion

To understand the sensitivity of the $D(\theta)$ transform, we have to analyze the "shape" features that are being captured. A convex/concave segment is mapped also to

a continuous curve in $\rho(t)$. The number of points in $\rho(\theta)$ does not depend directly on the length of the contour. It is instead proportional to the angular span of the curve, and the pattern of $\rho(\theta)$ depends on the curvature and position of the curve segments in the image plane. In the $D(\theta)$ transform, we compute the perpendicular distances between pairs of parallel tangents. It basically captures the relative position of a curve segment with respect to other curve segments of the contour and $\Delta\theta$ determines the sampling rate of the measurement. Changes in the $\rho(\theta)$ transform will be reflected in the $D(\theta)$ transform. There are basically two kinds of changes in $\rho(\theta)$: (i) displacement of the branches; and (ii) shortening or lengthening of branches. These changes corresponds to one or more of the following kinds of distortions of curve segments in the image domain.

1. small local distortions (bumpy contour),
2. changes in curvature,
3. changes in angular span,
4. changes in position (translation).

Because of thresholding (in computing the SLHT) and quantization of ρ, small local distortions will be filtered out to a certain extent.

For distortions that affect a significant angular span, the effects on $D(\theta)$ will be much bigger. To simplify the discussion, we assume that

1. The shape contour is composed of k segments and each segment has an angular span equal to 180°. Hence, there are k points in each column of $\rho(\theta)$, and $2k-3$ points in each column of $D(\theta)$. Here, we also neglect the intersecting points of the curves in $\rho(\theta)$.
2. One of the k segments is being distorted and the distortion affects the complete segment. Consequently, one of the k points in each column of $\rho(\theta)$ will be displaced.

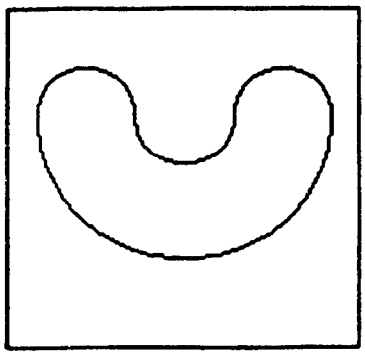
Let's consider one column of the transforms. If the distorted segment overlaps with the convex hull of the curve, then either the top most or the bottom most point in that column of $\rho(\theta)$ is displaced. Consequently, $k-1$ points in $D(\theta)$ will be affected. In the other case, some point in between the top most or the bottom most points of $\rho(\theta)$ is displaced, and only two points in $D(\theta)$ will be affected. This means that the $D(\theta)$ transform is more sensitive to distortions that affect the convex hull of the object. If $k-1$ points out of $2k-3$ points are displaced, then $(2k-3)-(k-1)$ points will match and $2(k-1)$ points ($k-1$ points of the test template and $k-1$ points of the reference template) will have a miss. Hence the score of the correlation (matching) function (neglecting the effects of quantization) is

$$\frac{2\times((2k-3)-(k-1))m_\theta - 2\times(k-1)m_\theta}{(2k-3)m_\theta}\times 100 = \frac{-2}{2k-3}\times 100$$

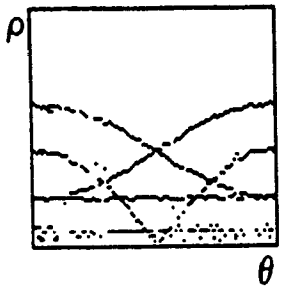
where m_θ is the number of quantized columns. For the other case, two points in each column of $D(\theta)$ will be displaced. Hence, the score of the matching function will be

$$\frac{2\times((2k-3)-2))m_\theta - 2\times 2m_\theta}{(2k-3)m_\theta}\times 100 = \frac{4k-14}{2k-3}\times 100.$$

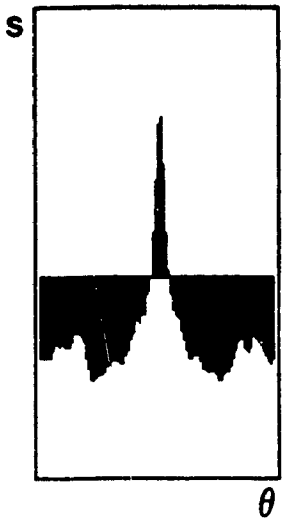
In real life, distortion of one curve segment will also affect its adjacent segments. Hence, the above two expressions may not give us very accurate estimates of the correlation score. Four examples of distortion and the matching scores are shown in figure 5.4. The shape shown in figure 3.1 is again used as the reference shape, and the shape in figure 5.4(a1) is used as a control sample for comparison. The matching score for figure 5.4(c) is better than the score predicted by the above expression. This is mainly because of the quantization effects. At the two ends of the bottom curve segment, the positions of the tangents are quite close to the non-distorted case. For the other two cases, figures 5.4(d) and 5.4(e), the score is lower than the score predicted. This is because the distortion affects all the three upper curve segments to different extents and part of these curves overlaps with the convex hull of the contour.



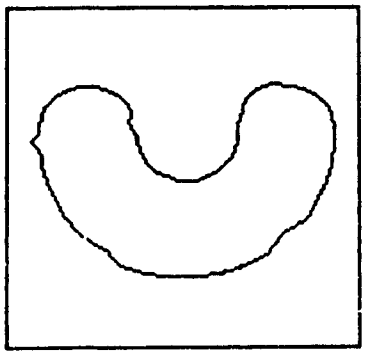
(a1) an instance of the shape shown in fig. 3.1;



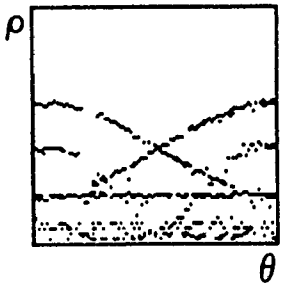
(a2) $D(\theta)$ transform of (a1);



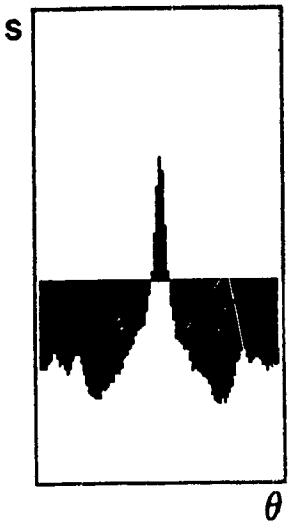
(a3) result of matching (a2) against fig. 5.1(b1), peak found at $(\theta, score) = (90°, 119)$;



(b1) an instance of (a1) with local distortion;

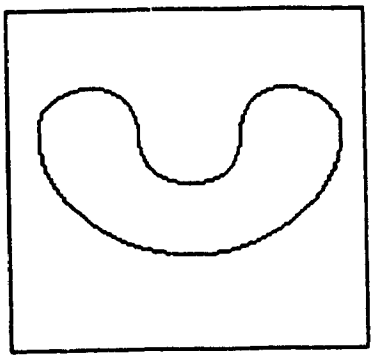


(b2) $D(\theta)$ transform of (b1);

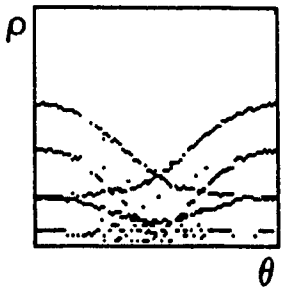


(b3) result of matching (b2) against fig. 5.1(b1), peak found at $(\theta, score) = (88°, 90)$;

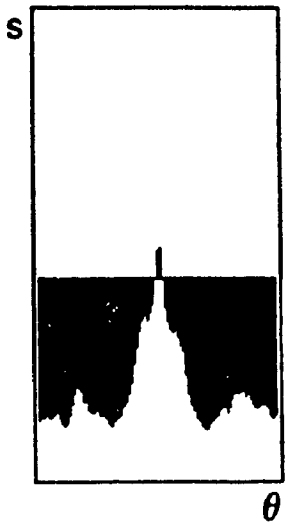
Figure 5.4 Recognition of distorted objects using the STIRS signature.



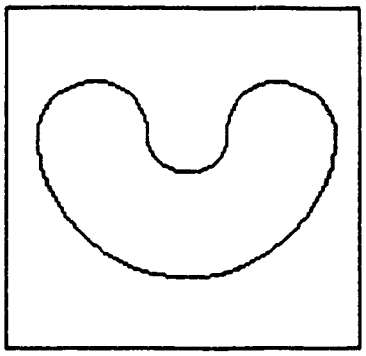
(c1) a distorted instance of (a1), the bottom curve is changed to an ellipse;



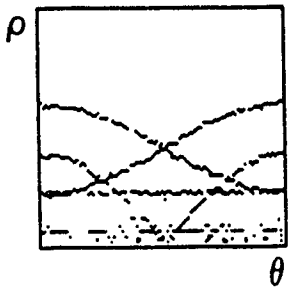
(c2) $D(\theta)$ transform of (c1);



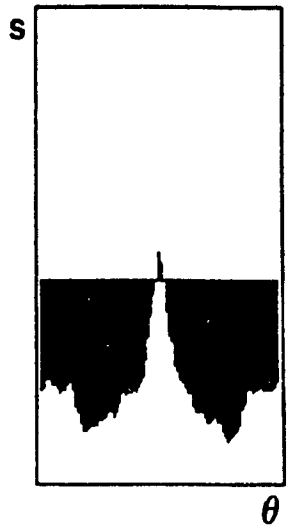
(c3) result of matching (c2) against fig. 5.1(b1), peak found at $(\theta, score) = (90°, 23)$;



(d1) a distorted instance of (a1), radius of the upper half-circles are modified;

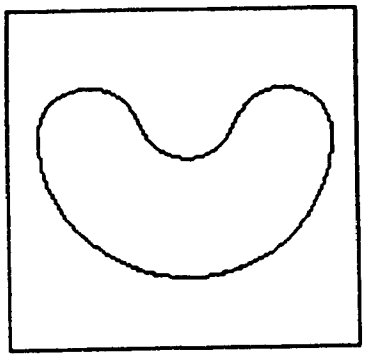


(d2) $D(\theta)$ transform of (d1);

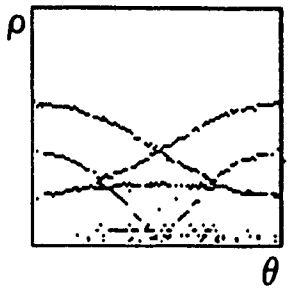


(d3) result of matching (d2) against fig. 5.1(b1), peak found at $(\theta, score) = (88°, 20)$;

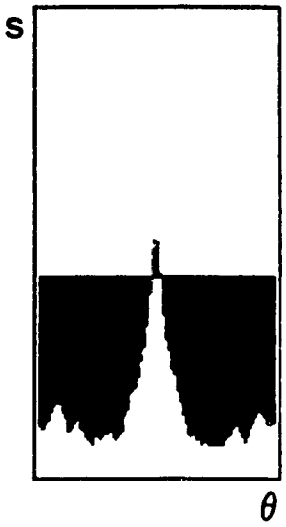
Figure 5.4 (Continued).



(c1) a distorted instance of (a1), the middle half-circle of the upper half-circles is shifted up and the angular span of its neighbor half-circles are reduced by 30°;



(c2) $D(\theta)$ transform of (c1);



(c3) result of matching (c2) against fig. 5.1(b1), peak found at $(\theta, score) = (90°, 23)$;

Figure 5.4 (Continued).

5.3. Remarks

Basically, the STIRS signature captures the characteristics of curvy segments of the object contour, such as angular span, curvature, and relative position with respect to other segments, and de-emphasizes straight edges. Because of the definition of the signature, it is very sensitive to distortion that affect a significant angular span, especially for the segments that coincide with the convex hull of the shape. However, small local distortions may be filtered out because of the smoothing effect of quantization. One way to avoid de-emphasizing straight edges is to assign weights to points in $D(\theta)$ according to the length of the line segments extracted in the SLHT, so that points in $D(\theta)$ that are due to long line segments will be given larger weights. This weighting scheme will improve the recognition accuracy of patterns that have long straight edges, such as the shape used in example 3.

In general, the $D(\theta)$ transform can be applied to images that are composed of a set of "smooth" curves which are not necessarily closed. This generalization will allow us to relax our assumption that the object have to be properly segmented. Partially occluded objects may also be recognized. Without a priori segmentation, we can partition the image into a number of smaller subimages. The $D(\theta)$ transforms of all the subimages are computed (partial signatures) and matched against the reference. If there is sufficient similarity between the partial signature and the reference, then we combine the subimage with its neighbors and compute the new partial signature. To do that, we need only to add the translation term to the corresponding SLHT's of the subimages and then compute the new partial signature. The merging of the subimages is accepted if the new partial signature has a better match with the reference. Merging of subimages stop when we arrive at a predefined degree of similarity or when the size of the subimage is larger than or equal to the size of the reference object.

Potential applications of this method include matching of maps and fingerprints. A previous attempt to apply Hough transform to extract ridges of fingerprints is

reported in [LinW83]. In fingerprint identification, the test image needs to be matched against a large number of reference images. Performing the matching in the $D(\theta)$ space is attractive because of the low matching cost. However, more in depth investigations on the distortion model of fingerprints, noise, and quantization errors are needed before we can affirmatively conclude that the $D(\theta)$ transform is applicable to this problem.

Chapter 6

Extraction of Graphic Features for Automatic Conversion of Engineering Line Drawings

In this chapter, we consolidate the curve detection techniques presented in chapters 2 and 4 to extract graphic features from engineering line drawings. We will first give a brief introduction to the problem and then present our solution.

6.1. Introduction

With recent advances in and wide spread use of computer-aided design and drafting, there is a strong demand to convert large volumes of existing paper drawings into the CAD database in order to maximize the benefits of the computer technology [Hof86, Jos89, Kar85, Pfe85]. Three categories of drawings have so far received much attention.

1. Map and utility plan drawings [Ale87, Eji84, Hos86, Mae85].
2. Electrical circuit and logic diagrams [Ble84, Bun81, Fuk84, LinX85, Oka88].
3. Mechanical and architectural drawings [Bix85,88, Bla81, Cle81, Jos89, Nag90].

There are very distinct processing requirements for these three categories of drawings. In the case of map and utility plan drawings, the images consist of straight lines, arbitrary curves and text annotation. Automatic conversion involves the extraction of text and an accurate approximation of the region boundaries, roads, and rivers, such that queries on the length of a road joining two cities, and area of a region can be answered. Hence, line tracking and curve fitting techniques serve well in this

application.

For circuit/logic diagrams, there is a set of predefined symbols and the information to be extracted is the symbol identities and the connectivity of wires among them. Exact locations and size of symbols, and parametric representation of the wires are not important. Hence, this kind of drawings are best processed by line tracking together with pattern recognition techniques.

A mechanical drawing is a geometric description of an artifact. It is mostly composed of parametric curves and text annotation. Extraction of the geometric features (parametric curves) in mechanical drawings is very different from finding an approximation to the curves as in the case of map processing. For example, we can approximate a curve by a set of straight lines. But this representation is not adequate as in editing and deriving intelligent interpretation.

Conversion of mechanical drawings is essentially a computer vision problem. The ultimate goal is not only to produce a compact encoding of the paper drawing in electronic format, but also to infer an intelligent interpretation of the geometric information of the object being drawn such that integration with CAM (computer-aided manufacturing) systems becomes possible [Bob85, Cho85]. There are active research projects on various aspects of interpretation of mechanical engineering drawings, such as recognition of dimensioning information [Ant90, Dor89 & 90], reconstruction and modification of 2D models from dimensioning information [Lig82, Wat88], reconstruction of 3D models from multiple 2D views [Bin86, Ric86, Rus87, Sak83]. All these papers assume that a "perfect" feature set is available. A "perfect" feature set is the set of features, text and graphics, that a draftsman uses to describe the drawings as if he were creating the drawings using a CAD station. However, the graphic features extracted by line tracking and curve fitting methods are far from perfect in this sense. In this chapter, we present a new approach to extract geometric features from mechanical part drawings.

6.1.1. The Graphic Features Extraction (GFE) Problem

A mechanical parts drawing consists of text (alphanumeric characters and mathematical symbols) and graphics. The graphics can be broadly divided into two categories: (i) line-structured entities, e.g. straight lines and curves; and (ii) special graphic symbols, e.g. arrowheads, machining symbols, etc. Each line-structured entity is associated with

1. *geometric attributes*: parametrization of the curve such as end points of a straight line, center and radius of a circle, etc.;
2. *display attributes*: such as thickness, solid or dashed, and dash length and gap width;
3. *semantic attributes*: the meaning of the graphic entity, e.g. visible contour, dimensioning, hidden edge, etc.

The semantic attributes are to be derived from the display attributes and context (its neighboring graphic entities and the text annotation). For example, thick solid lines are usually used to represent the visible contour of the object, whereas thin solid lines can be used as dimensioning, sectioning, folding lines and other purposes. To infer the geometric model of the drawing, we need correct recognition of both the geometric and semantic attributes of the graphic entities. Our view of the GFE problem is essentially a segmentation problem.

1. *Text/Graphic (T/G) segmentation*

   Given a set of data points S, we want to find a partition $S = T \cup G$ where G is the set of data points due to the line-structured entities, and T is the set of data points due to the text in the drawings including the special graphic symbols. If the text touches the graphics, then $T \cap G \neq \emptyset$.

2. *Line-structured graphic entities extraction*

   Given the set of graphic data points G, we want to infer a set of curves C

(together with the geometric, display and semantic attributes) such that (i) G can be reconstructed from C, within some prespecified maximum allowable error and (ii) the representation we obtain is as close to the "perfect" feature set as possible. If we can segment G into a number of sets of pixels, and each set of pixels corresponds to a curve, then it won't be difficult to compute the best geometric and display attributes of each curve. To derive the semantic attributes of the curves, we need to recognize other graphic symbols, such as arrowheads, and the inference process will be context sensitive.

In our work, we are mainly concerned with the extraction of geometric and display attributes of line structured graphic entities.

6.1.2. Previous Work

In the literature so far, T/G segmentation and graphic features extraction are treated separately [Ant90, Ble84, KaR90a]. The output of the T/G segmentation is used as input to the graphic feature extraction module. In the absence of any contextual information and without explicitly recognizing the characters, the only properties that can be made use of in T/G segmentation are the character sizes, and spacings of characters [Ble84, Fle88]. With such limited available information, the segmentation fails in the following situations:

1. text touching graphics (most of the time, text touches straight lines in the drawing, e.g. guidelines, tables, dimension lines),
2. isolated single or double characters,
3. occasions where dashed lines are confused as text string.

In order to overcome these problems, we need to be able to extract the graphic entities in the presence of text strings, and in turn make use of the extracted information to guide the segmentation of the text strings. For example, the first problem can be solved

by first extracting the lines that touch the text before we carry out the T/G segmentation.

The approach commonly used to extract line-structured graphic entities is based on local processing, such as line tracking and curve fitting [Bla81, KaR90b, Pav82]. In order to fit a curve, some initial partition (segmentation) of the set of graphic data points G is assumed. Usually, the critical points along the chain of pixels, corners, and junction points are used to divide the curves into a number of segments. Individual segments are fitted with straight lines; if the fitting is not accepted, then the segment is fitted with higher order curves such as conic [Cle81, Nag90] or cubic splines [Bix88]. The acceptance criteria of the fitting are based on some local error measures. A "best" fit is obtained by an iterative split-and-merge process to merge adjacent connected segments to form larger curves. There are several drawbacks to this bottom-up approach.

1. An adverse side effect of the split-and-merge approach is that a segment can be part of a single curve only. Consequently, if a curve intersects or touches other lines/curves, it will be fragmented into pieces and individual pieces may be fitted by curves not necessarily of the same type.
2. Since only local information is used, the fitting process is sensitive to local distortions that are introduced during preprocessing, e.g. thinning.
3. This method also fails to recognize dashed curves. For example, individual segments of a dashed circle are very well fitted by straight lines. However, when all the segments of the dashed circle are considered collectively, then the group of segments is best represented by a circle.

6.2. The Graphic Features Extraction System

To overcome the above mentioned problems, we need to make use of some global information to aid the decision on segmentation, and we should also allow pixels

to possess multiple memberships (belonging to more than one curves). The Hough transform is applied to generate initial hypotheses (collective evidence) of the presence of curves in the transform space, and we verify the hypotheses in the image space. The segmentation of the image is refined iteratively. The requirement on a priori segmentation is, therefore, relaxed. Three different kinds of hypotheses are considered in this work: (i) solid long straight lines; (ii) circles; and (iii) dashed lines.

Although the Hough transform is a robust method to detect parametric curves, direct implementation of the method outlined in chapter 4 will be computationally expensive. To speedup the processing, we adopt a stepwise segmentation refinement strategy. Note that most of the curves in the image are connected to other curves by long straight lines. If we can logically remove those long straight lines, then the image can be divided into a number of smaller blocks, which contains only a small number of curves. Then we can apply the curve detection technique to extract the solid circles in individual blocks. Line segments in θ–ρ corresponding to logically removed pixels will not take place in subsequent hypotheses generations, but the removed pixels will still be taken into account in the hypotheses verification process (this is to allow for multiple memberships of pixels). In other words, the dominant graphic features are successively extracted, and the segmentation of the residue image is further refined after each step. The classes of graphic features will be extracted in the following order: solid long straight lines, solid circles, dashed circles concentric with previously extracted circles, dashed lines, dashed circles and finally shorter straight lines.

6.2.1. Implementation Details and Experimental Results

Throughout the discussion in this section, we assume that the drawings follow the ANSI standard [Gia78]. The input image is digitized with a resolution of 300 points per inch. The test image (after thinning) shown in figure 6.2a (on page 122) is used as the example in the following discussion. The size of the test image is 1536×1536.

The processing steps are summarized below. Our emphasis is on segmentation rather than on deriving the best parameters of the curves. The experimentation is only aimed at demonstrating the feasibility of the approach.

Step 1: Long straight lines extraction.

We first compute the SLHT with $\Delta\rho = 2$ units, $\Delta\theta = 1°$, and $L_{min} = 10$ units. The SLHT is regarded as a sampling tool in this application which samples the image in different orientations and collects evidence of the presence of straight lines. The following two criteria are used to determine if a line segment is a true straight line or not. In this step, we are only interested in extracting the long straight lines

1. if the length of the segment is larger than some threshold, L_{long}, or
2. if the length of the segment is larger than some smaller threshold, L_{nor}, and there is no curvy extension from either ends of the segment. Referring to figure 6.1, if the line segment $l_1 = (\theta_1, \rho_1)$ is part of a curve, then, there exists some line segment $l_2 = (\theta_2, \rho_2)$ in adjacent columns[8] such that (i) l_2 overlaps with l_1, (ii) l_2 extends beyond the end points of l_1, and (iii) length of l_2 is $\geq$ 1/3 the length of l_1. If we trace along the same direction, we can find some other segment l_3 which overlaps with l_2, extends beyond l_2 and with length $\geq$ 1/3 the length of l_1. Hence, if there exists a trace of overlapping segments, l_1, l_2, ..., l_k such that the angle between l_k and l_1 is at least, say 10°, and the length of each segment in the list is at least 1/3 the length of l_1, then we conclude that l_1 is not a straight line.

The second criterion allows us to deduce the presence of straight lines with average length. In actual implementation, we need to take care of the possible quantization

[8] The θ–ρ plane is considered to be circular and the $(m_\theta - 1)$–th column is said to be adjacent to the 0-th column.

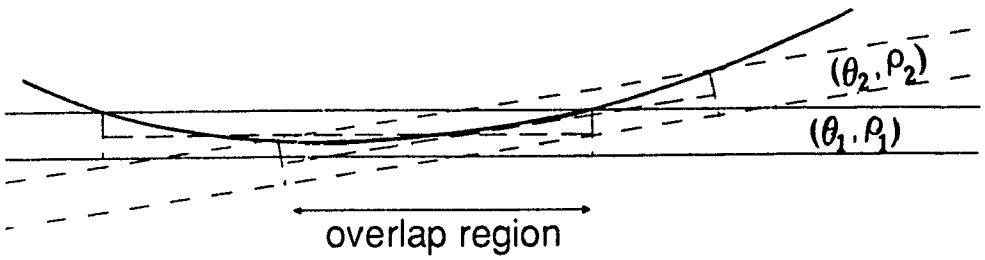


Figure 6.1 Sampling of a curve segment.

errors [Van81]. Hence, we take into consideration the possible spreading of line segments across adjacent buckets when determining the length of the line segments.

In this sampling process, it is possible to have multiple selected line segments which correspond to the same straight line. An effective representation of the group of segments is obtained as follows:

Lines Selection Algorithm2

threshold the accumulator array and collect line segments whose lengths are $\geq L_{nor}$;
FOR each line segment in the list whose length is $\leq L_{long}$ check for curvy extension;
group together segments in the list that overlap with one another;

FOR each group that does not contain any segment marked "curve" do
  find the longest segment in the group, say $l_1 = (\theta_1, \rho_1)$;
  compute the bounding boxes with orientations $\theta_1 \pm 0.5°$, θ_1, and $\theta_1 \pm 1°$
    that enclose the end points of all the segments in the group;
  select the narrowest bounding box as the representative;
  IF the width of representative bounding box is less than $4 \times \Delta\rho$ then
    accept it as a hypothesis of a straight line;

In the experiment we set L_{long} to 200 and L_{nor} to 65. In the process of verification and removal of the straight lines, we want to preserve the connectivity of the other curves that run across the lines. To achieve this, we first find the junction points in the image and assign to those points a label equal to 3. Then the foreground pixels in the 7 by 7 neighborhood of a junction point is assigned a label equal to 2. Other foreground pixels will have labels equal to 1. To verify the hypothesis, we open up a rectangular window of width $4\times\Delta\rho$ in the image plane corresponding to the hypothesis. We start from the mid point of the hypothesis and scan outwards. The foreground pixels lying under the scanning strip will be deleted by assigning a label of -1, provided there is no foreground pixels under the scanning strip having labels larger than 1. The scanning stops when we encounter a block of four consecutive background pixels. Figures 6.2b and 6.2c shows the segmentation of the image after this step.

Step 2: Solid circles extraction.

We first compute the connected components of the remaining foreground pixels and label the line segments in $\theta-\rho$ accordingly. To label the line segments in $\theta-\rho$, we inspect three pixels of the segment, the two end points and the mid point[9]. If the three pixels have the same label, then the segment will be assigned the value of that label, otherwise, the segment is assigned a label of -1. Line segments corresponding to removed pixels will be assigned negative labels. All segments with negative labels are discarded. The circle detection module is then applied to individual components with 250 or more pixels. There are seven components in figure 6.2c with 250 or more pixels. In performing the voting, we set $\Delta t = \Delta\rho$, $\Delta\alpha = \Delta\theta$, and $TA_{thre} = 200$. The time complexity of the voting process is proportional to k^3 where k is the average number

[9] Because of quantization, the mid point obtained from the two end points may not correspond to a foreground pixel. This is overcome by performing a search on the 3 by 3 neighborhood of the mid point for foreground pixels.

of points in each column of the compact boolean θ–ρ array. The value of t_s is set according to the value of k such that t_s is interpolated between 1 to 5 for k ranging from 8 or above down to 1.

The peaks detected in t–α are only rough estimates of the center of potential circles. Depending on the value of t, an error in α, say 0.25°, may result in a displacement of 5 to 10 pixels in the image plane. So we perform a simple search around the detected value of α in steps of 0.25° and compute the 1D histogram for radius estimation. The value of α that results in the sharpest and highest peak in the 1D histogram is selected.

For each hypothesis of a circle, we open up a ring shaped window in the image plane and scan along the ring. The run length code of the foreground pixels (including the previously removed pixels) along the scanning ring is computed and small patches (of size less than or equal to 3) of background pixels is filtered away. Also, patches of foreground pixels of size smaller than 5 are discarded. These patches correspond to other lines/curves that run across the ring. Foreground pixels lying under the accepted patches are removed except for junction points and their neighbors. Figure 6.2d shows the circles extracted in this step. After removing the foreground pixels, we also clean up the θ–ρ space by discarding the segments that correspond to the removed pixels.

Step 3: Extraction of concentric dashed circles.

In mechanical drawings, it is very common to have dashed circles drawn concentric to solid circles. So we can make use of the center coordinates of the circles found in the previous step as initial hypotheses. The compact boolean θ–ρ array is reinitialized to include all the remaining line segments in θ–ρ. To find the hypotheses of dashed circles, we simply compute a 1D histogram for radius estimation using the center coordinates of extracted solid circles. If peaks are found in the histogram, then we will further verify the hypotheses as in the case of solid circle extraction. Figure

6.2e shows the dashed circles found in this step. At this state of the system development, no further analysis is implemented to recognize the dashed patterns of the dashed circles.

Step 4: Dashed lines extraction.

Up to this stage, long straight lines, solid curves and concentric circles are extracted. The most dominant graphic features in the residue image is the dashed lines. Three different types of dashed lines are considered in this work:

1. *hidden line:* short-short-short [-short]
2. *center line:* long-short-long [-short-long]
3. *phantom line:* long-short-short-long [-short-short-long]

The options in square brackets mean arbitrary number of repetitions. The following rules are used in the recognition of dashed lines:

1. Length of short segments are less than 50 units.
2. Length of long segments are at least 40 units.
3. The ratio of the lengths of long segments to short segments in a dashed line is at least 2.
4. The gap between segments should not be larger than the length of the short segments in the dashed line.

The choice of the line length limits is to allow for possible variations in the drawing practice of different people. Collinear lines in the image plane will be mapped to adjacent buckets in the θ–ρ space. Hence, we can generate hypotheses of dashed lines by finding clusters of line segments (having distinct labels) in θ–ρ. Previously extracted long straight lines may also be part of a dashed line, for example, the long segments of center lines and phantom lines. So we will insert the extracted long lines into the θ–ρ space, but they are given labels -2, -3 and so on to distinguish them from

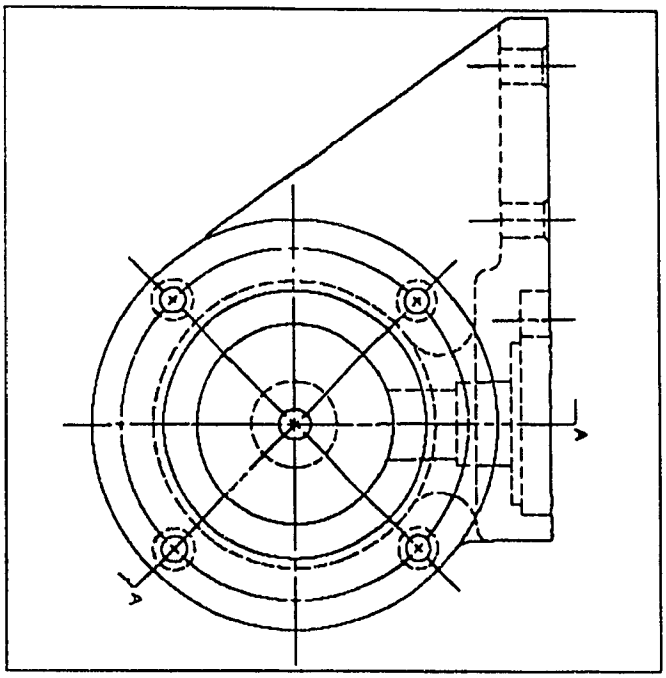
the other line segments in θ–ρ. To verify the dashed line hypothesis, we compute the effective representation of the cluster of line segments, i.e. the narrowest bounding box that encloses the segments. A rectangular window is opened up in the image plane and the run length code of the patches of foreground pixels are analyzed. The three different types of dashed lines are recognized by finite state automata. The extracted dashed lines are shown in figure 6.2f.

Step 5: Extraction of remaining dashed circles.

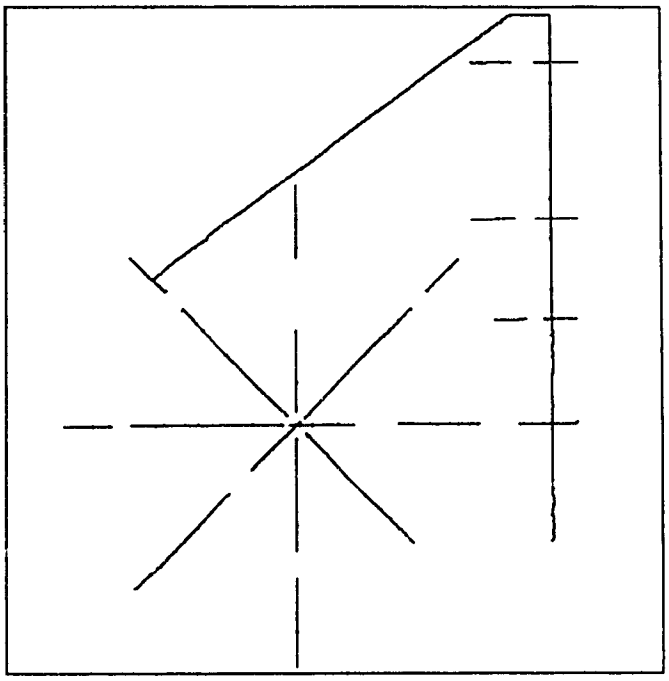
At this stage, we clean up the θ–ρ space and perform yet another iteration of circle detection. All the remaining line segments in θ–ρ will be taking part in this step. Noting that the remaining dashed circles are likely to be partial circles, and the number of segments in θ–ρ will be relatively small, we increase the number of samples taken in the voting process by setting t_s to 8 and lower the value of TA_{thre} to 100 in the peak detection process. Figure 6.2g shows the dashed circles found in this step.

Step 6: Polygonal approximation of remaining objects.

Objects in the residue image (figure 6.2h) are to be approximated by straight lines. There are a number of small crosses in the image which are due to the junction preservation strategy. These crosses can be filtered away by counter checking with the extracted graphic entities. This step has not been implemented.

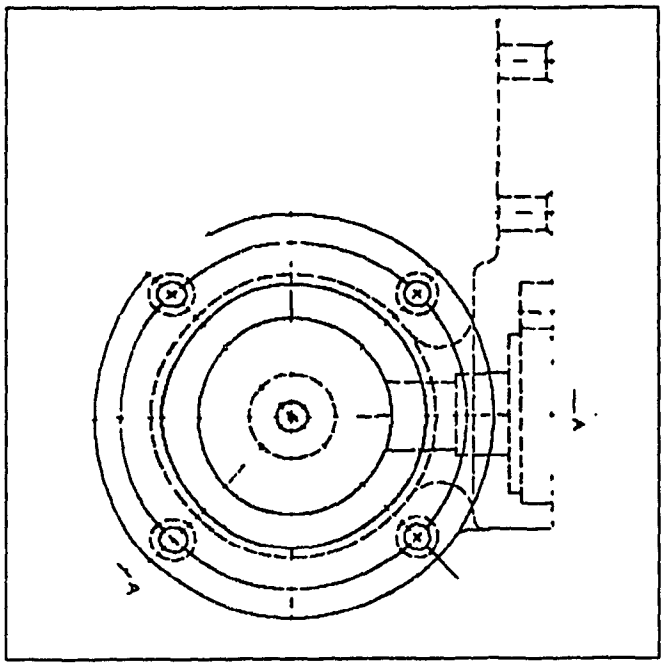


(a) The input test image after thinning;

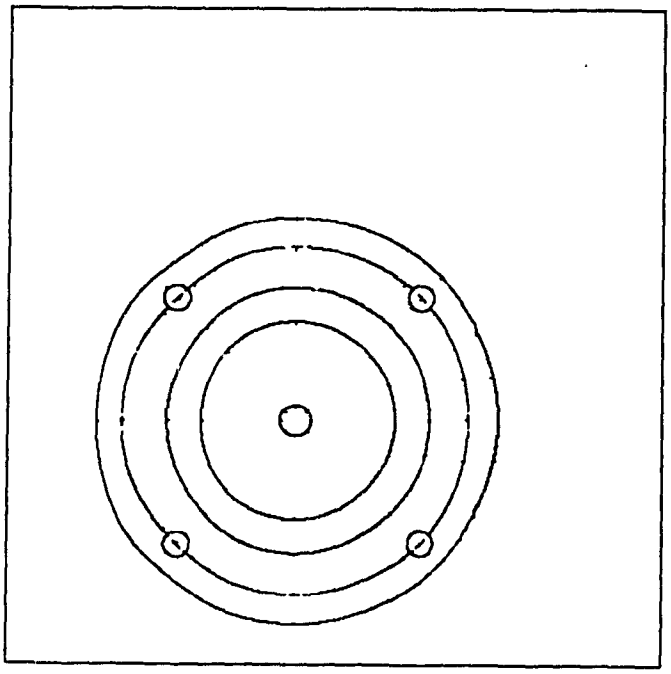


(b) The extracted long straight lines;

Figure 6.2 Example 1 of graphic features extraction.

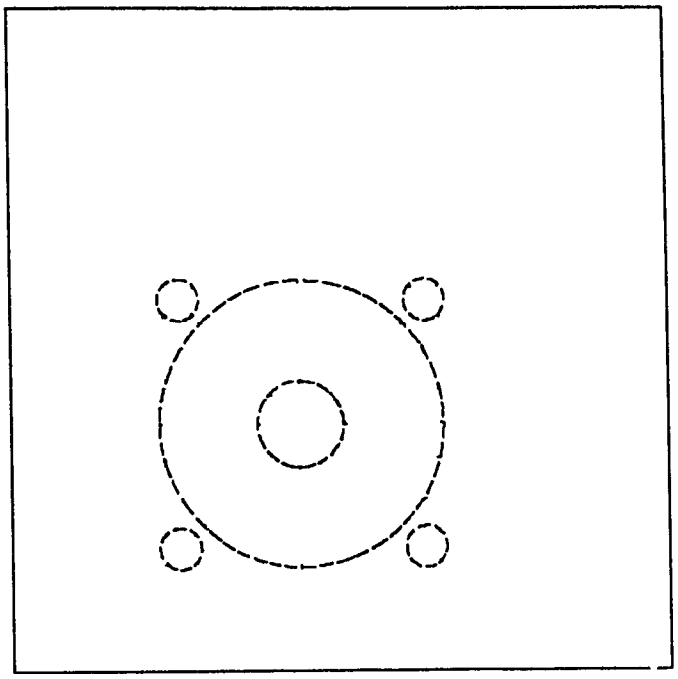


(c) The segmented image after removal of long straight segments.

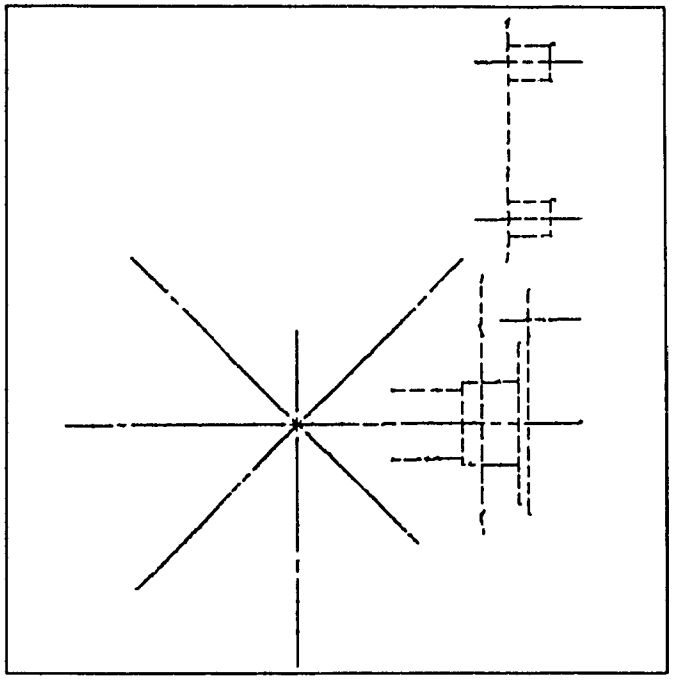


(d) The extracted circles;

Figure 6.2 (Continued).

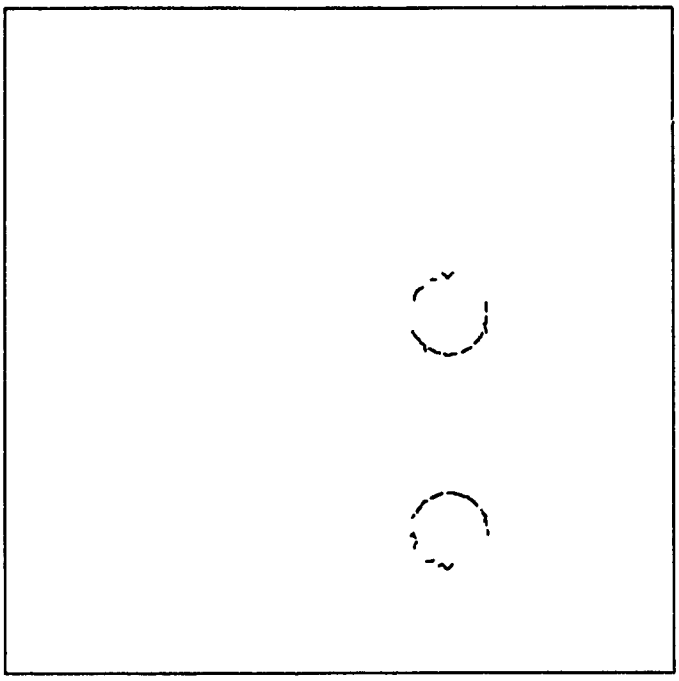


(e) The extracted concentric dashed circles;

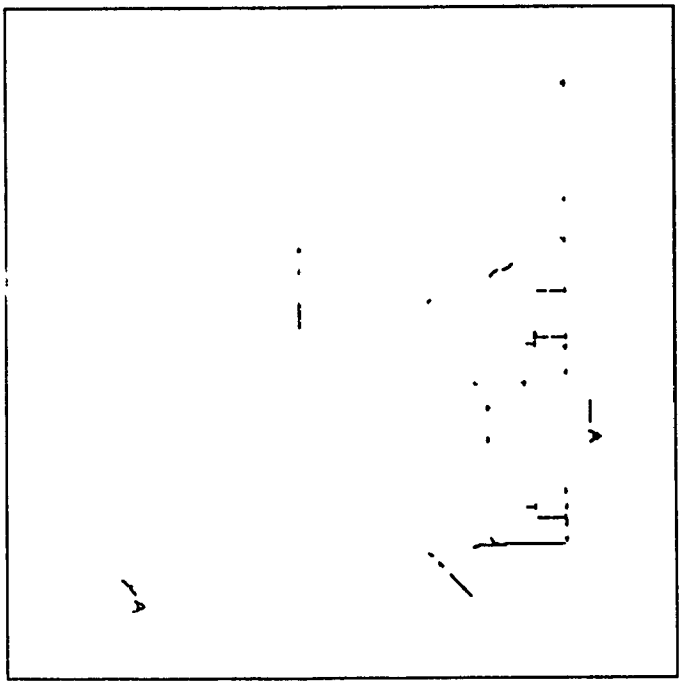


(f) The extracted dashed lines;

Figure 6.2 (Continued).



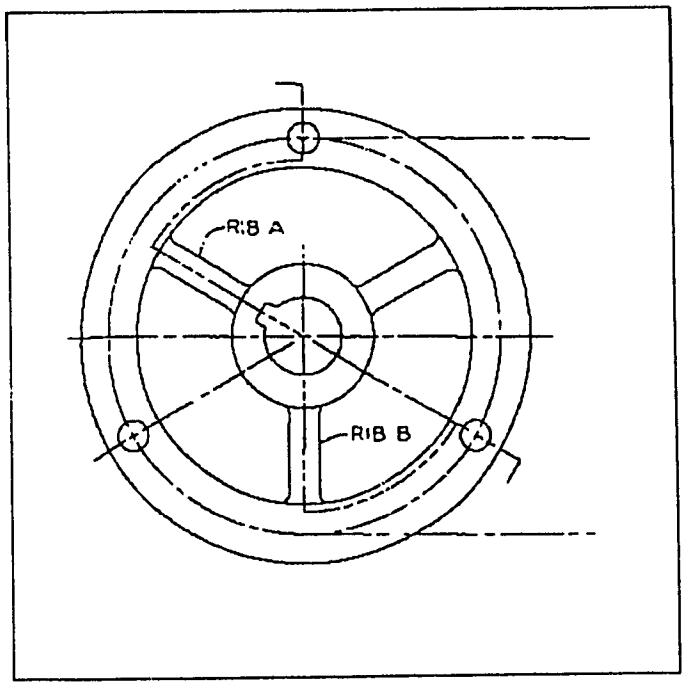
(g) The extracted dashed circles;



(h) The residue components;

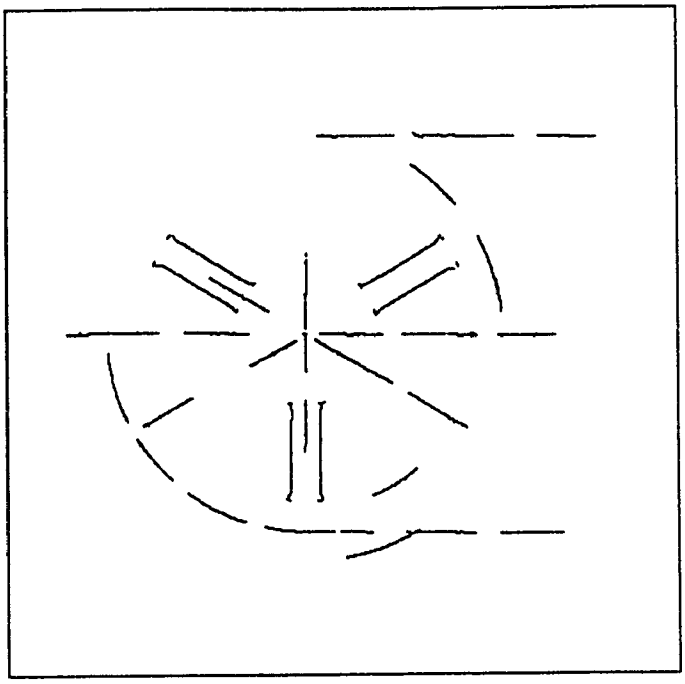
Figure 6.2 (Continued).

Another example is shown in figure 6.3. The size of the second test image is 1024×1024. Several segments of the dashed circle are classified as straight lines in step 1. However, these segments are reclassified as a dashed circle in step 3 when more global information is taken into consideration. Some of the dashed line segments are not classified correctly. This is due to imperfect image quality and digitization. Some of the long segments of the center lines are broken and the gap between short segments is longer than the length of the segments. In the above two examples, although the results are shown in the form of binary images, the extracted entities are being represented in parametric format.

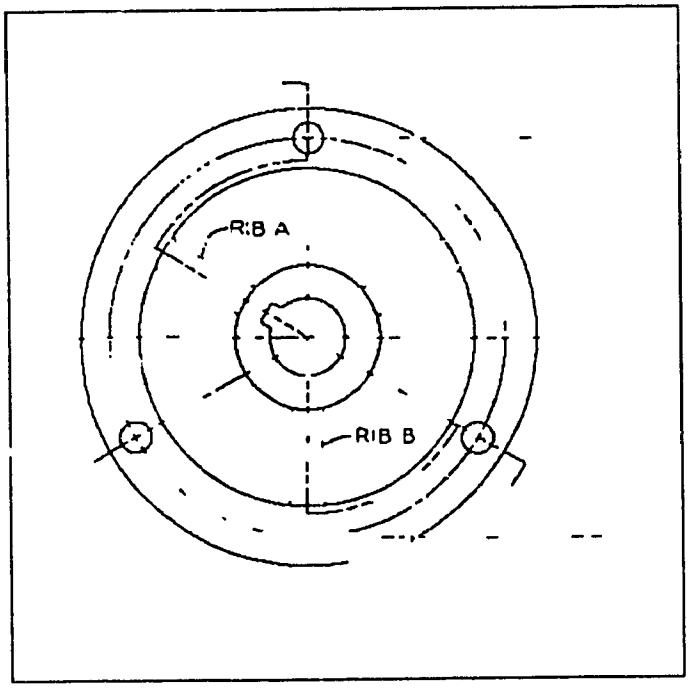


(a) The input test image after thinning;

Figure 6.3 Example 2 of graphic features extraction.

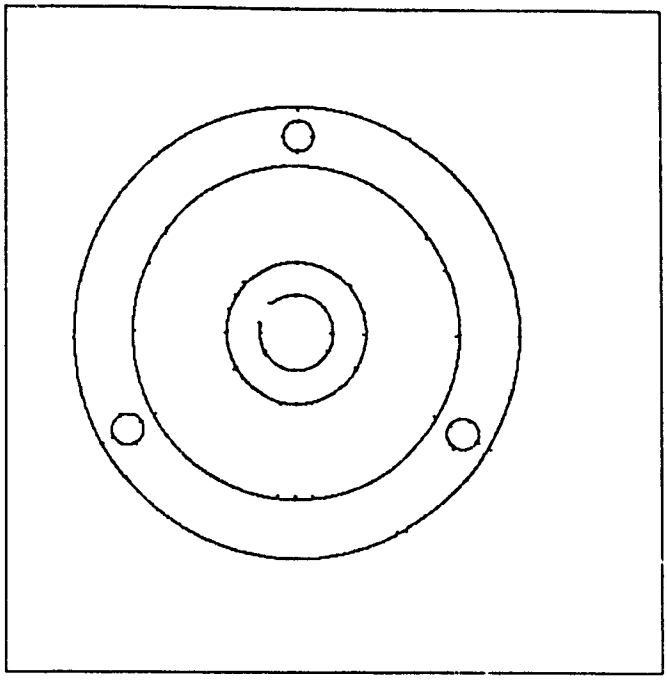


(b) The extracted long straight lines;

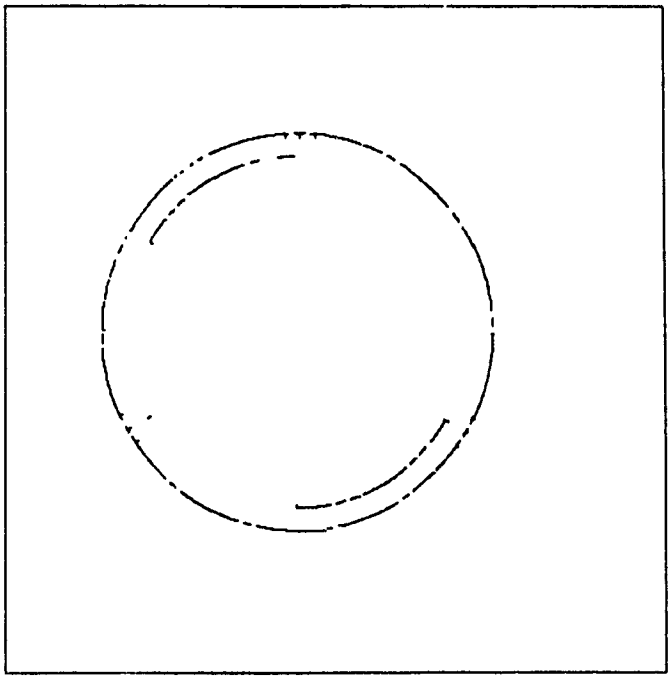


(c) The segmented image after removal of long straight segments.

Figure 6.3 (Continued).

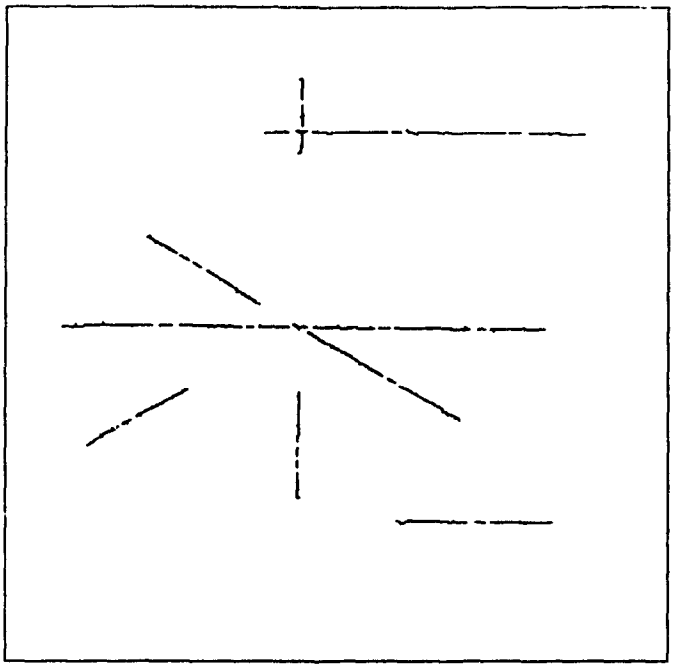


(d) The extracted circles;

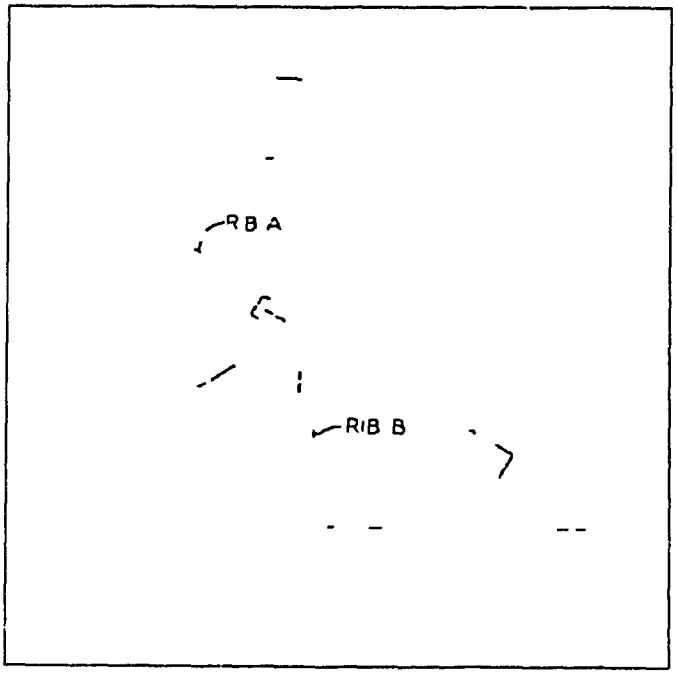


(e) The extracted concentric dashed circles;

Figure 6.3 (Continued).



(f) The extracted dashed lines;



(g) The residue components;

Figure 6.3 (Continued).

6.2.2. Comments on the Current System

The objective of the experiment is to demonstrate the feasibility of the approach. Besides speed, there are a number of aspects that can be further improved. For example,

1. It is desirable to replace the simple verification process by a more sophisticated module that will at the same time compute the best parametric representation of the curve. The little crosses in the residue image may also be avoided.
2. Hypotheses of other kind of features, such as ellipses and sectioning (set of equally spaced parallel lines), can be generated in the θ–ρ space.
3. The rules set out in the previous subsection for dashed lines recognition are inadequate to deal with poor image quality and nonconformity with the drawing conventions. Developing more dedicated interpretation techniques that possess certain form of drawing correction capability to deal with poor image quality and nonconformity with drawing conventions is a challenging research problem.

6.3. Remarks

Hough transform is a computationally expensive method. The tradeoff in this application is the quality of the extracted features and potential gains in both speed and functional performances of the overall system (when integrated with other modules) and in higher level processing. From the experimentation, good segmentations are achieved with this global processing technique. Moreover, the method is amenable to parallel implementation. Hence, the computation time can be shortened considerably if parallel machines are available.

In real life system development, it is desirable to further convert the extracted graphic features into some standard CAD format, such as the IGES standard [Smi83], such that the converted drawings can be edited using commercially available CAD

systems. In our work, we only considered the extraction of line-structured graphic entities, however, our system can be easily integrated with other functional modules, such as the text/graphic segmentation module and the graphic symbols recognition module. Our opinion on T/G segmentation is that it should consist of multiple phases (steps) and interact with the graphic extraction module. For example, the T/G segmentation can be divided into three major steps:

1. Initial segmentation of blocks of text using modified run length coding technique [Wah82].
2. Extraction of long text strings using the regularity properties of lettering [Fle88]. This step should be executed after long straight lines and solid circles are extracted from the image, so that text characters are separated from graphics.
3. Extraction of short text strings, such as single and double characters, will require certain amount of contextual information, such as if there are character strings, center lines, and arrowheads, etc. in the vicinity. So this step is preferably executed after center lines and phantom lines and arrowheads are recognized.

Chapter 7
Concluding Remarks

In this thesis, we studied the use of Hough transform to detect 2D curves in binary images. Improvements to both the functional performance and computation efficiency are presented. We have shown that by incorporating connectivity check in the detection of straight lines, spurious lines can be avoided. The method is more robust against random noises. Isolated noise and unrelated pixels of far away objects are filtered out. Moreover, the extracted features (line segments) carry certain syntactic information, such as the end points of the lines and the normal parameters of the tangent to the curve. Knowing the end points of the line segments allows us to replace the thresholding process by a more refined line selection process. Depending on the application requirements, we can obtain a linearization of the image or select the long straight lines. This makes the modified Hough transform a more powerful feature extractor.

Another important aspect of Hough transform is its inherent parallelism. A systolic architecture that implements the modified Hough transform for straight lines detection has been developed. Our architecture has superior area-time complexity compared with other parallel implementations. In systolic computation, the processing elements only communicate synchronously with their neighbors. This implies that the computation only involves local data dependencies and exhibits regular synchronization, which are the prerequisites of any form of parallel processing. Systolic algorithms can be easily mapped to other parallel architectures, such as the SIMD array processors or MIMD multiprocessor systems. Hence, our systolic algorithm need not be

restricted to VLSI implementation.

Tangents are useful features in object recognition. Novel techniques for detecting circles and ellipses, and recognizing arbitrary smooth curves using the set of tangents have been developed. One major advantage of formulating the transform using the tangent representation is that it allows a natural way to decompose the high dimensional parameter space into three subspaces, namely, the translation, rotation and intrinsic parameters. Parameters of the three subspaces can be determined separately, thus huge savings in both time and space are achievable. Moreover, our methods are very robust against random noise. This is because in computing the SLHT, isolated noise pixels will be filtered out.

Our methodology requires that the SLHT of the image be computed. This may represent considerable computation overhead. However, it is not uncommon in computer vision applications that straight lines are also important features to be extracted. In these cases, the SLHT will be computed anyway and already available for use to the curves detection algorithms. In chapter 4, we analyzed the asymptotic complexities of different Hough transform algorithms for detecting ellipses. A comprehensive simulation study on the computational efficiencies and functional performances of the Hough transform algorithms is an interesting topic for further research.

Although we are not the first to investigate recognition methodologies using the set of tangents, in this thesis we established the theoretical foundation of these methodologies by proving the uniqueness of the SLHT transform of closed smooth curves. There are, however, two issues worth further research. The first one is concerned with the sensitivity of the method. During our simulation studies, we observed that tangents from different ellipses will interfere with each other and may lead to the formation of spurious peaks in the t--α space. However, the degree of interference tends to decrease when the two ellipses are further apart. More research on the behaviour of the voting algorithms is desirable. Comparing the two methods for ellipses detection presented in

presented in chapter 4, the second method is more computationaly efficient than the first one. However, the first method is more robust in detecting dashed curves, and the second method tends to suffer more from the sensitivity problems.

The second issue is related to the ability of the STIRS signature in distinguishing shapes. Formal characterization of the distinctness of the signature with respect to variations of shapes will help to determine the reliability and applicability of the method to real life problems. Two potential applications of the STIRS signature are matching of maps and fingerprints. It is worthwhile to conduct further investigation into these two problems.

The overall orientation of the thesis is geared towards theoretical development and less emphasis is placed on applications. This is partly due to the author's personal interests and partly because of the author's lack of solid experiences in real life computer vision applications. In chapter 6, we consolidated the curves detection techniques and applied it to extract graphic features for automatic conversion of engineering line drawings. There are two major reasons for the choice of that application as a pilot study. First, engineering line drawings are mostly composed of parametric curves. Second, the drawing rules are relatively well documented. In this application, the ultimate goal is to infer an intelligent interpretation of the objects being drawn. Therefore, correct recognition of the feature classes is more important than the accuracy to which the extracted features approximate the input image. The feature extraction problem is formulated as a segmentation problem. The justification of using Hough transform in this application is its ability to detect curves without a priori segmentation. Hypotheses of the presence of curves are generated in the transform space, and then verified in the image space. The whole image is taken into account and more reliable decisions on segmentation can be made. One characteristic of our method is that the classification of the features at a stage is only tentative. This classification can be overridden in latter stages when more global information is taken into account. More

reliable features are extracted using this integrated approach of segmentation and feature extraction.

Automatic conversion of engineering line drawings is by no means a simple problem. Our contribution is only to one of the many aspects of the overall problem. A lot more research and development are needed before we can actually build a complete line-drawing processing system.

References

[Ale87] J. Alemany and R. Kasturi, "A computer vision system for interpretation of paper-based maps", Proc. SPIE, 829, Applications of Digitial Image Processing, pp 125-137 (1987).

[Ant90] D. Antoine, S. Collin and K. Tombre, "Analysis of technical documents: the REDRAW system", Workshop on Syntactic and Structural Pattern Recognition, pp 1-20 (1990).

[Anu85] Paul Edward Anuta, "Parameter space techniques for temporal image registration", Ph.D. Dissertation, Purdue University (1985).

[Bal81] D.H. Ballard, "Generalizing the Hough Transform to Detect Arbitrary Shapes", Pattern Recognition, 13, pp 111-122 (1981).

[Bal82] D. H. Ballard and C. M. Brown. Computer Vision. Prentice-Hall, 1982.

[Ban88] Amit Bandopadhay and Jung Liang Fu, "Searching parameter spaces with noisy linear constraints", Conf. Computer Vision and Pattern Recognition, pp 550-555 (1988).

[Ben66] Russell V. Benson. Euclidean Geometry and Convexity. McGraw-Hill, 1966.

[Bin86] Ho Bin, "Inputting constructive solid geometry representations directly from 2D orthographic engineering drawings", CAD, 18(3), pp 147-155 (1986).

[Bix85] J. P. Bixler, J. P. Sanford, "A technique for encoding lines and regions in engineering drawings", Pattern Recognition, 18(5), pp 367-377 (1985).

[Bix88] J. P. Bixler, L. T. Watson and J. P. Sanford, "Spline-based recognition of straight lines and curves in engineering line drawings", Image and Vision Computing, 6(4), pp 262-269 (1988).

[Bla81] W. Black, T. P. Clement, et al., "A general purpose follower for line-structured data", Pattern Recognition, 14, pp 33-42 (1981).

[Ble84] Heinrich Bley, "Segmentation and preprocessing of electrical schematics using picture graphs", CVGIP, 28, pp 271-288 (1984).

[Bob85] James E. Bobrow, "NC machine tool path generation from CSG part representations", CAD, 17(2), pp 69-76 (1985).

[Bro83] C. M. Brown, "Inherent bias and noise in the Hough transform", IEEE Trans. PAMI, 5, pp 493-505 (1983).

[Bun81] H. Bunke, "Automatic interpretation of lines and text in circuit diagrams", in Pattern Recognition Theory and Applications, J. Kittler et al. (ed.), pp 297-310 (1981).

[Cas87] D. Casasent and R. Krishnapuram, "Curved object location by Hough transformations and inversions", Pattern Recognition, 20, pp 181-188 (1987).

[Cha74] G. D. Chakerian. "A characterization of curves of constant width". American Mathematical Monthly, 81, pp 153-155 (1974).

[Che89] Fang-Hsuan Cheng, Wen-Hsing Hsu and Mei-Ying Chen, "Recognition of handwritten Chinese characters by modified Hough transform techniques", IEEE Trans. PAMI, 11(4), pp 429-439 (1989).

[Cho85] B. K. Choi, M. M. Barash, "STOPP: an approach to CADCAM integration", CAD, 17(4), pp 162-168 (1985).

[Chu85] Henry Y.H. Chuang, and C.C. Li, "A Systolic Array Processor for Straight Line Detection by Modified Hough Transform", Proc. Comp. Arch. for Pattern Anal. & Database Management, pp 300-303 (1985).

[Cle81] T. P. Clement, "The extraction of line-structured data from engineering drawings", Pattern Recognition, 14(1), pp 43-52 (1981).

[Cow83] A. E. Cowart, W. E. Snyder and W. H. Ruedger, "The detection of unresolved targets using the Hough transform", CVGIP, 21, pp 2[illegible]3 (1983).

[Cyp87] R. Cypher, J. L. C. Sanz and L. Snyder, "The Hough transform has $O(N)$ complexity on SIMD $N \times N$ mesh array architectures", 1987 Workshop on Computer Architectures for Pattern Analysis and Machine Intelligence, pp 115-121 (1987).

[Dav84] R. Davis and D. Thomas, "Systolic array chip matches the pace of high-speed processing", Electronic Design, pp 207-218 (Oct. 1984).

[Dor89] Dov Dori, "A syntactic/geometric approach to recognition of dimensions in engineering machine drawings", CVGIP, 47, pp 271-291 (1989).

[Dor90] Dov Dori, "Self structural syntax directed pattern recognition of dimensioning components in engineering drawings", Workshop on Syntactic and Structural Pattern Recognition, pp 88-112 (1990).

[Dud75] R.O. Duda and P.E. Hart, "Use of the Hough transformation to detect lines

and curves in pictures", Comm. of ACM, 15, pp 11-15 (1975).

[Dye83] C.R. Dyer, "Gauge inspection using Hough transform", IEEE Trans. PAMI, 5, pp 621-623 (1983).

[Eji84] M. Ejiri et al, "Automatic recognition of design drawings and maps", Int. Conf. Pattern Recognition, pp 1296-1305 (1984).

[Eng88] Jan R. Engelbrecht and Friedrich M. Wahl, "Polyhedral Object Recognition Using Hough-Space Features", Pattern Recognition, 21, pp 155-167 (1988).

[Fis89] A. L. Fisher and P. T. Highnam, "Computing the Hough transform on a scan line array processor", IEEE Trans. PAMI, 11(3), pp 262-265 (1989).

[Fle88] L. A. Fletcher and R. Kasturi, "A robust algorithm for text string separation from mixed text/graphics images", IEEE Trans. PAMI, 10, pp 910-918 (1988).

[Fuk84] Youji Fukada, "A primary algorithm for the understanding of logic circuit diagrams", Pattern Recognition, 17(1), pp 125-134 (1984).

[Ful81] M. C. Fulford, "The Fastrak automatic digitizing system", Pattern Recognition, 14, 65-74 (1981).

[Ger87] G. Gerig, "Linking image-space and accumulator-space: a new approach for object-recognition", IEEE 1st Int. Conf. Computer Vision, pp 112-117 (1987).

[Gia78] J. W. Giachino and H. J. Beukema, "Engineering technical drafting 4th edition", American Technical Society, 1978.

[Gri90] W. Eric L. Grimson and Daniel P. Huttenlocher, "On the sensitivity of the Hough transform for object recognition", IEEE Trans. PAMI, 12(3), pp 255-274 (1990).

[Gue89] Concettine Guerra and Susanne Hambrusch, "Parallel algorithms for line detection on a mesh", J. of Parallel and Distributed Computing, 6, pp 1-19 (1989).

[Han87] K. Hanahara, T. Maruyama, T. Uchiyama, "A real-time processor for the Hough transform", IEEE Trans. PAMI, 10, pp 121-125 (1987).

[Hin90] S. C. Hinds and J. L. Fisher, "A document skew detection method using run-length encoding and the Hough transform", 10th ICPR, Vol. 1, pp 464-468 (1990).

[Hof86] J. Hofer-Alfeis, "Automatic conversion of existing mechanical-engineering drawings to CAD data structures: state of the art", IFIP Conf. CAPE'86, pp 259-267 (1986).

[Hor89] Alan Horwitz, "Reconstructing a function from its set of tangent lines", American Mathematical Monthly, 96(9), pp 807-813 (Nov. 1989).

[Hos86] T. Hoshino, S. Suzuki and M. Kosugi, "Automatic input method for large scale maps", 8th Int. Conf. on Pattern Recognition, pp 449-453 (1986).

[Hou62] P.V.C. Hough, "Method and means for recognizing complex patterns", U.S. Patent 3069654, 1962.

[Hua85] K.Y. Huang, K.S. Fu, T.H. Sheen, S.W. Cheng, "Image processing of seismograms: (A) Hough transformation for the detection of seismic patterns; (B) Thinning process in the seismogram", Pattern Recognition, 18, pp 429-440 (1985).

[Hwa84] Kai Hwang and Faye A. Briggs, Computer Architecture and Parallel Processing, McGraw-Hill (1984).

[Ill87] J. Illingworth, J. Kittler. "The adaptive Hough transform". IEEE Trans. PAMI, 9(5), pp 690-697 (1987).

[Ill88] J. Illingworth, J. Kittler. "A survey of the Hough transform", Computer Vision, Graphics and Image Processing, 44(1), pp 87-116 (1988).

[Ini84] R.M. Inigo, E.S. McVey, B.J. Berger, M.J. Wirtz, "Machine vision applied to vehicle guidance", IEEE Trans. PAMI, 6, pp 820-826 (1984).

[Jay83] S. N. Jayaramamurthy and R. Jain, "An approach to the segmentation of textured dynamic scenes", CVGIP, 21, pp 239-261 (1983).

[Jos89] S. H. Joseph, "Processing of engineering line drawings for automation input to CAD", Pattern Recognition, 22, pp 1-11 (1989).

[Kar85] M. Karima, K. S. Sadhal and T. O. McNeil, "From paper drawings to computer aided design", IEEE Comput. Graphics Applications, 5(2), pp 27-39 (1985).

[KaS83] S. Kasif, L. Kitchen and A. Rosenfeld, "A Hough transform technique for subgraph isomorphism", Pattern Recognition Letters, 2, pp 83-88 (1983).

[KaR90a] R. Kasturi et al., "A system for interpretation of line drawings", IEEE Trans. PAMI, 12(10), pp 978-991 (1990).

[KaR90b] R. Kasturi et al., "Document image analysis an overview of techniques for graphics recognition", Workshop on Syntactic and Structural Pattern Recognition, pp 192-230 (1990).

[Kri87a] R. Krishnapuram and David Casasent, "Hough space transformations for discrimination and distortion estimation", Computer Vision, Graphics and Image Processing, 38, pp 299-316 (1987).

[Kri87b] R. Krishnapuram, "Hough-space associative processor for pattern recognition", Ph.D. Dissertation, Carnegie Mellon University (1987).

[Kri89] R. Krishnapuram and D. Casasent, "Determination of three-dimensional object location and orientation from range images", IEEE Trans. PAMI, 11, pp 1158-67 (1989).

[Ku87] Kuo-Lung Ku, "Algorithms and architectures for the Hough transformation", Ph.D. Dissertation, University of Washington (1987).

[Kun82] H. T. Kung, "Why systolic architectures?" IEEE Computer, 15(1), pp 37-46 (1982).

[KuSY88] S. Y. Kung, VLSI Array Processors, Prentice Hall, 1988.

[Kus85] M. Kushnir, K. Abe, K. Matsumoto, "Recognition of hand-printed Hebrew characters using features selected in the Hough transform space", Pattern Recognition, 18, pp 103-113 (1985).

[Lea87] V. F. Leavers and J. F. Boyce, "The Radon transform and its application to shape parameterization in machine vision", Image and Vision Computing, 5, pp 161-166 (1987).

[Lee89] Andrew John Lee, "Optical processing techniques for satellite pose determination", Ph.D. Dissertation, Carnegie Mellon University (1989).

[Li86a] Hungwen Li, Mark A. Lavin, Ronald J. Le Master. "Fast Hough transform: a hierarchical approach", Computer Vision, Graphics and Image Processing, 36, pp 139-161 (1986).

[Li86b] H. Li, M. A. Lavin, "Fast Hough transform based on the bintree data structure", Conf. Computer Vision and Pattern Recognition, pp 640-642 (1986).

[LPJ89a] H. F. Li, Derek Pao, R. Jayakumar, "Improvements and systolic implementation of the Hough transformation for straight line detection", Pattern Recognition, 22(6), pp 697-706 (1989).

[LPJ89b] H. F. Li, Derek Pao and R. Jayakumar, "Systolic arrays: present state and issues", Int. Symp. Computer Arhitecture and Digital Signal Processing, pp 69-74 (1989).

[Lig82] Robert Light and David Gossard, "Modification of geometric models through variational geometry", CAD, 14(4), pp 209-214 (1982).

[LinW83] W. C. Lin and R. C. Dubes, "A review of ridge counting in dermatoglyphics", Pattern Recognition, 16, pp 1-8 (1983).

[LinX85] Xinggang Lin, Shigeyoshi Shimotosuji, Michihiko Minoh, "Efficient diagram understanding with characteristic pattern detection", CVGIP, 30, pp

84-105 (1985).

[Mae85] A. Maeda and J. Shibayama, "Application of automatic drawing reader for the utility management system", IEEE Workshop CAPAIDM, pp 139-141 (1985).

[McK90] D. S. McKenzie and S. R. Protheroe, "Curve description using the inverse Hough transform", Pattern Recognition, 23 (3/4), pp 283-290 (1990).

[Mea80] Carver Mead and Lynn Conway, Introduction to VLSI Systems, Addison-Wesley, 1980.

[Mer75] P.M. Merlin and D.J. Farber, 'A parallel mechanism for detecting curves in pictures", IEEE Trans. Computers, 24, pp 96-98 (1975).

[Mil86] V. J. Milenkovic, "Multiple resolution search techniques for the Hough transform in high dimensional parameter spaces", in Techniques for 3-D Machine Perception, A. Rosenfeld ed., Elsevier Science, 1986, pp 231-254.

[Mol82] D. I. Moldovan, "On the analysis and synthesis of VLSI algorithms", IEEE Trans. Computers, 31(11), pp 1121-1126 (1982).

[Mur88] K. Murakami, H. Koshimizu and K. Hasegawa, "An algorithm to extract convex hull on $\theta-\rho$ transform space", Int. Conf. Pattern Recognition, pp 500-503 (1988).

[Nag90] V. Nagasamy and N. A. Langrana, "Engineering drawing processing and vectorization system", CVGIP, 49, pp 379-397 (1990).

[Nib88] Wayne Niblack and Dragutin Petkovic, "On improving the accuracy of the Hough transform: theory, simulations and experiments", Conf. Computer Vision and Pattern Recognition, pp 574-579 (1988).

[Oka88] Akio Okazaki et al., "An automatic circuit diagram reader with loop-structure-based symbol recognition", IEEE Trans. PAMI, 10(3), pp 331-341 (1988).

[Pao90a] Derek Pao, H. F. Li and R. Jayakumar, "Detecting parametric curves using the straight line Hough transform", Int. Conf. Pattern Recognition, Vol 1, pp 620-625 (June 1990).

[Pao90b] Derek Pao, H. F. Li and R. Jayakumar, "Characterization and recognition of 2D smooth curves using the set of tangent lines", submitted to IEEE Trans. PAMI.

[Pao91] Derek Pao, H. F. Li and R. Jayakumar, "Graphic features extraction for automatic conversion of engineering line drawings", 1st Int. Conf. on Document Analysis and Recognition, in press (Sept. 1991).

[Pav82] T. Pavlidis, "Curve fitting as a pattern recognition problem", 6th Int. Conf. Pattern Recognition, pp 853-859 (1982).

[Pfe85] W. Pferd III, "Entering drawings into CADDS with the Skantek automatic digitizer - methods and results", Proc. CAMP'85, pp 263-269 (1985).

[Poe86] W. Poelzleitner, "A Hough transform method to segment images of wooden boards", Int. Joint Conf. on Pattern Recognition, pp 262-264 (1986).

[Pri90] J. L. Prince and A. S. Willsky, "Reconstructing convex sets from support line measurements", IEEE Trans. PAMI, 12(4), pp 377-389 (1990).

[PIK90] John Princen, John Illingworth and Josef Kittler, "A hierarchical approach to line extraction based on the Hough transform", CVGIP, 52, 57-77 (1990).

[Rad86] C. J. Radford, "Optical flow fields in Hough transform space", Pattern Recognition Letters, 4, pp 293-305.

[RiJS89] J. S. Richards, "Real-time optical Hough transform and morphological inspection techniques", Ph.D. Dissertation, Carnegie Mellon University (1989).

[Ric86] T. H. Richards and G. C. Onwuboln, "Automatic interpretation of engineering drawings for 3D surface representation in CAD", CAD, 18(3), pp 156-160 (1986).

[Roc70] R. T. Rockafellar, Convex Analysis. Princeton University Press, 1970.

[Rus87] C. L. Russon, "Integrating 3D modelling and 2D drafting", CADCAM'87 Conf., pp 233-236 (1987).

[Sak83] H. Sakurai, D. C. Gossard, "Solid model input through orthographic views", Comp. Graphics, 17(3), pp 243-252 (1983).

[Sha75] S. D. Shapiro, "Transformations for the computer detection of curves in noisy pictures", CGIP, 4, pp 328-338 (1975).

[Sha78] S. D. Shapiro, "Properties of transforms for the detection of curves in noisy images", CGIP, 8, pp 219-236 (1978).

[Sha79] S. D. Shapiro and A. Iannino, "Geometric constructions for predicting Hough transform performance", IEEE Trans. PAMI, 1, pp 310-317 (1979).

[Shu87] D.B. Shu, C.C. Li, F. Mancuso, Y.N. Sun, "A Line Extraction Method for Automated SEM Inspection of VLSI Resist", IEEE Trans. PAMI, 10, pp 117-120 (1987).

[Sil85] T.M. Silberberg, "The Hough transform on the Geometric Arithmetic Parallel Processor", IEEE Workshop on Computer Architecture for Pattern

Analysis and Image Database Management, pp 387-393 (1985).

[Smi83] B. M. Smith et al, Initial Graphics Exchange Specification (IGES), version 2.0, U.S. Department of Commerce, NBSIR 82-2631 (AF), Feb. 1983.

[Tay90] Russell W. Taylor, "An efficient implementation of decomposable parameter space", 10th ICPR, Vol. 1, pp 613-619 (1990).

[Tsuj78] S. Tsuji, F. Matsumoto. "Detection of ellipses by a modified Hough transform", IEEE Trans. Computers, 27, pp 777-781 (1978).

[Tsuk83] H. Tsukune, Keisuke Goto, "Extracting elliptical figures from an edge vector field", Conf. Computer Vision and Pattern Recognition, pp 138-141 (1983).

[Van81] T.M. van Veen and F.C.A. Groen, "Discretization errors in the Hough transform", Pattern Recognition, 14, pp 137-145 (1981).

[Wah82] F. M. Wahl, M. K. Y. Wong and R. G. Casey, "Block segmentation and text extraction in mixed text/image documents", CVGIP, 20, pp 375-390 (1982).

[Wal85] R.S. Wallace, "A modified Hough transform for lines", Conf. Computer Vision and Pattern Recognition, pp 665-667 (1985).

[Wat88] S. Watson, "Relational geometry - a new generation of two-dimensional CAD", Computer-Aided Eng. Journal, 5(4), pp 169-172 (1988).

[Yam81] S. Yam, L. S. Davis, "Image registration using generalized Hough transform", Conf. Pattern Recognition and Image Processing, pp 526-533 (1981).