Towards an Optimal Accumulator Size For the Hough Transform

by

Mohammad Wasim Akhtar

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

ELECTRICAL ENGINEERING

August, 1991

Order Number 1354066

Towards an optimal accumulator size for the Hough Transform

Akhtar, Mohammad Wasim, M.S.

King Fahd University of Petroleum and Minerals (Saudi Arabia), 1991

U·M·I

300 N. Zeeb Rd.
Ann Arbor, MI 48106

TOWARDS AN OPTIMAL ACCUMULATOR SIZE FOR THE HOUGH TRANSFORM

BY

MOHAMMAD WASIM AKHTAR

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

ELECTRICAL ENGINEERING

AUGUST 1991

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

This thesis, written by

MOHAMMAD WASIM AKHTAR

under the direction of his thesis committee, and approved by all the members, has been presented to and accepted by the Dean, College of Graduate Studies, in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

Thesis Committee

A. Zaman.

Chairman (Dr. Mohammad Atiquzzaman)

Member (Dr. Nasser Tabbayeb)

Member (Dr. Syed Zaki Al-Akhdar)

Member (Dr. Khalid Hussain Biyari)

S.A Al-Baiyat

Dr. Samir Alwan Al-Baiyat
Department Chairman

Dr. Ala H. Rabeh
Dean College of Graduate Studies

Date : 24-9-91

Dedicated

to my

Parents & Brothers

ACKNOWLEDGEMENT

Praise and gratitude be to Allah the Almighty, with whose gracious help it was possible to accomplish this work. Acknowledgement is due to King Fahd University of Petroleum and Minerals for extending all the facilities and providing financial support.

I would like to express my sincere appreciation and gratefulness to my advisor and committee chairman Dr. Mohammad Atiquzzaman. His patient guidance, innovative suggestions and continuous encouragement were of vital help in achieving this goal. I am also thankful to my committee members Dr. Mammar Bettayeb, Dr. Syed Zaki Al-Akhdar and Dr. Khalid Hussain Biyari for their critical suggestions and the invaluable co-operation extended by them.

Special thanks are to Dr. Gerhard F. Beckhoff for his help and co-operation in expediting the completion of the thesis.

Last, but not the least, I am thankful to all the faculty, staff, colleagues and friends who made my stay at K.F.U.P.M a memorable and valuable experience.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

A	:	two dimensional accumulator array
$a_{i,j}$	:	cell in the accumulator array with i as the index in the ρ – direction and j as the index in the θ – direction
b_a	:	apperent width of the line segment L
$b_{i,j}$	:	bar in the image plane corresponding to $a_{i,j}$
C_k	:	k^{th} column in A
c	:	intercept of the line segment on the y-axis
$d_{j,k}$	:	difference in the indexes of the columns C_j and C_k in the accumulator array
d_m	:	half the maximum width of L
f_{same}	:	number of feature points along L having one of the co-ordinates equal
g_x	:	gray level gradient in the x-direction
g_y	:	gray level gradient in the y-direction
h	:	grid constant in Freeman's grid intersection method
I	:	one dimensional array for projection
L	:	line segment in the image plane
l_{actual}	:	actual length of L
l_{ave}	:	length of L averaged over several columns of A

l_{max}	:	maximum length of L
l_t	:	nearest straight line
l_1	:	first likely straight line or the feedback straight line
l_2	:	second likely straight line
m	:	slope of L
$m_{predicted}$	:	slope of L determined from $\theta_{predicted}$
N	:	image size
N_1	:	number of feature points along l_1
n_{connec}	:	number of nonzero entries for which C_k is scanned between η^k_{sl} and η^k_{sh} to overcome the effect of noise
n_f	:	number of feature points in the image
$n_{i,j}$	:	number of votes or the count in $a_{i,j}$
n_1	:	estimated number of feature points on l_1 such that $n_1 > 0.7N_1$
n_2	:	estimted number of feature points on l_2
n^k_ρ	:	spread along the ρ − direction in C_k
(x_i, y_i)	:	co-ordinates of a feature point in the image plane in x and y directions
(x_{int}, y_{int})	:	co-ordinates of the point of intersection of the predicted line segment with the image boundaries
(x_1, y_1) & (x_2, y_2)	:	extremities of L

Greek Symbols

$\Delta\theta$	:	the resolution along the θ – axis of the accumulator array
$\Delta\rho$	:	the resolution along the ρ – axis of the accumulator array
ε_s	:	the error minimised for computing the slope-intercept parameters of l_2
ε_x	:	maximum error in the computed value of the x-co-ordinates of the extremities
ε_y	:	maximum error in the computed value of the y-co-ordinates of the extremities
η^k_{correc}	:	number of cells subtracted or added to η^k_{st} or η^k_{sh} as correction
η^k_{fin}	:	index in the ρ – direction corresponding to the first non-zero entry in C_k
η^k_{lin}	:	index in the ρ – direction corresponding to the last non-zero entry in C_k
η^k_{sh}	:	index of the cell in the ρ – direction whose corresponding bar in the image plane has $\rho = \rho^k_{starth}$
η^k_{st}	:	index of the cell in the ρ – direction whose corresponding bar in the image plane has $\rho = \rho^k_{startl}$
θ	:	angle of the normal to L w.r.to x-axis
θ_{asize}	:	size of A in the θ – direction

θ_i	:	angle of the normal to l_i
θ_k	:	angle of the normal to the bars corresponding to cells in C_k
θ_{max}	:	maximum value of θ
θ_{min}	:	minimum value of θ
$\theta^{CHT}_{predicted}$	:	predicted value of θ using CHT
θ_s	:	sampled value of θ
μ_o	:	average number of pixels along L
ρ	:	length of the normal from origin to L
ρ_{axis}	:	size of A in the ρ – direction
ρ_{exact}	:	exact value of ρ ($\rho_{exact} = \rho_{min} + \rho_{peak} \Delta\rho$)
ρ_{high}	:	quantized value of ρ on the ρ – axis such that $\rho_{high} = \rho_{min} + (\rho_{peak} + 1)\Delta\rho$
ρ_j	:	length of the normal to l_j
ρ_{low}	:	quantized value of ρ on the ρ – axis such that $\rho_{low} = \rho_{min} + (\rho_{peak} - 1)\Delta\rho$
ρ_{max}	:	maximum value of ρ
ρ_{min}	:	minimum value of ρ
$\rho^{CHT}_{predicted}$	:	predicted value of ρ using CHT
ρ^k_{starth}	:	the value of ρ of the bar corresponding to the cell upto which scanning is done
ρ^k_{startl}	:	the value of ρ of the bar corresponding to the cell from which scanning is started

ρ_1^k	:	length of the normal corresponding to the first non-zero entry in C_k
ρ_2^k	:	length of the normal corresponding to the last non-zero entry in C_k
σ_o	:	standard deviation of the distribution of the feature points around L in the orthogonal direction to L
τ_c	:	threshold for the count in a cell of A
τ_g	:	threshold for discontinuties in L
τ_p	:	threshold applied to A for detecting multiple line segments
$(\rho_{actual}, \theta_{actual})$	:	actual parameters of L
$(\rho_{computed}, \theta_{computed})$	:	actual parameters of L computed from its extremities
$(\rho_{peak}, \theta_{peak})$	:	indexes in parameter space corresponding to the peak in A
$(\rho_{predicted}, \theta_{predicted})$	:	parameters computed from the peak in A
(ρ_i, θ_i)	:	parameters of l_i

THESIS ABSTRACT

NAME OF STUDENT : **MOHAMMAD WASIM AKHTAR**

TITLE OF STUDY : **TOWARDS AN OPTIMAL ACCUMULATOR SIZE FOR THE HOUGH TRANSFORM**

MAJOR FIELD : **ELECTRICAL ENGINEERING**

DATE OF DEGREE : **AUGUST, 1991**

Hough Transform (HT) is widely used in Computer Vision. HT is a mapping from the image plane onto the parameter space. It can detect any shape which can be defined by a parametric equation. When HT is used to detect straight line segments in images, the conventional approach only gives information regarding the parameters defining the line segment. It does not give the exact position of the line segment. Some authors have developed procedures to obtain the complete line segment description using the HT, but their methods are all highly compute-bound and require large storage space. In this thesis a new approach for obtaining the complete line segment description from the information in the accumulator array is developed and a criterion for an optimal accumulator size is suggested. The developed approach reduces the computational complexity of the HT. The new approach has been tested on synthetic images without noise, synthetic images with noise, and also on real world images which inevitably contain noise.

MASTER OF SCIENCE

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

Dhahran, Saudi Arabia

بسم الله الرحمن الرحيم

خلاصة البحث

عنوان البحث : نحو مجمع ذي قياس أمثل لمحول [Hough]

اسم الطالب : محمد وسيم أختر

التخصص : هندسة كهربائية

تاريخ الدرجة : صفر ١٤١٢ / أغسطس ١٩٩١

إن [Hough Transform] ,HT, صار يستخدم كثيراً اليوم في مجال استخدام الحاسب الآلي للرؤية و التصوير . و إن هذا المحول هو عبارة عن نقل شيء ما من مجال الصورة إلى مجال التغيرات و هو يستطيع أن يتبين أي شكل يمكن أن يعرَّف بمعادله ما و لكن هذا النوع من المحولات عند استخدامه لاستبيان قطع من خطوط مستقيمة في الصورة فإن الطريقة التقليدية لا تعطي إلا معلومات ضئيلة متعلقة بالمتغيرات التي تعرف تلك القطعة فقط، بالاضافة إلى أنها لا تستطيع أن تحدد مكان القطعة المستقيمة من الصورة بشكل دقيق.

و مؤخراً اقترح بعض الباحثين طرقاً ليحصل على وصف كامل لتلك القطع المستقيمة في الصورة و ذلك باستخدام HT أيضاً. و لكن هذه الطريقة المقترحة عليها مآخذ من وجهين : الأول أنها تحتاج إلى وقت طويل للحسابات ثم هي بحاجة إلى ذاكرة كبيرة.

في هذا البحث، أقترحت طريقة جديدة تستطيع أن تعطي وصفاً كاملاً عن الصورة و ذلك من خلال المعلومات المتوفرة في مصفوفة المجمع. و تم اقترح عدة معايير من أول الوصول لأمثل قياس لهذا المجمع. و تم تطبيقه على نماذج مختلفة من الصوره المشومة و غير المشومة بالإضافة إلى صوره حقيقية و كلها أثبتت فعالية هذه الطريقة المقترحة. و من مزايا هذه الطريقة على غير ها من الطرق أنها قللت بدرجه كبيرة صعوبة الحاسابات الموجودة في HT.

درجة الماجستير في العلوم
جامعة الملك فهد للبترول و المعادن
الظهران - المملكة العربية السعودية
صفر ١٤١٢ هـ - أغسطس ١٩٩١م

1. INTRODUCTION

Of the five senses - vision, hearing, smell, taste and touch - vision is undoubtedly the one that man has come to depend upon above all others and indeed the one that provides most of the data he receives. A important feature of the human vision is that each glance provides megabits of information and data rates for continuous viewing exceed several megabits per second. Another feature of the human visual system is the ease and speed with which interpretation of a scene is carried out. However, the important point is that we are for the most part unaware of the complexities of vision. Man is inventive and is now trying to get machines to do much of his work for him. But vision for a machine is difficult compared to that for a human eye.

Machine vision is achieved through Object Recognition - which is necessarily a problem of discrimination i.e., discriminating between patterns of different classes. During the period prior to 1970, image processing and statistical pattern recognition consumed much research effort and effectively dominated the image analysis [1]. This is one of the reasons why Hough Transform was not given due recognition in research and was not optimized adequately. Hough Transform was first proposed in a patent by P.V.C.Hough in 1962 [44] and later developed by Duda and Hart in 1972 [2].

The field of image processing and object recognition may be broadly divided into three processing stages : [1]

1. Low - level processing

2. Intermediate - level processing

3. Application - level processing

1.1 Low - level Processing

The low - level processing stage of image processing and object recognition involves :

- basic imaging operations such as convolution, FFT etc;
- image filtering techniques such as median filtering, mode filtering, and noise suppression by Gaussian smoothing.
- Thresholding techniques like image enhancement, image segmentation, and edge detection.

1.2 Intermediate - level Processing

At this stage, the primary importance is of obtaining abstract information

about images rather than converting an image into another which is done in low - level processing stage. This stage of machine vision involves use of one or several of the following shape detection algorithms (choice of which depends on the nature of the image) :

-- Line detection

-- Circle detection

-- Ellipse detection

-- Polygon detection

-- Hole and Corner detection

The shape detection processes listed above can be effectively carried out by using Hough Transform.

1.3 Application - level Processing

The high - level or the application - level processing is the last stage and consists of :

-- Abstract pattern matching techniques

-- Automated visual inspection

– Statistical pattern recognition

– Real-Time electronic hardware systems

1.4 Motivation behind the work

As stated before, Hough Transform falls in the intermediate - level processing stage. It has been widely used for detection of parametric shapes. Its adoption in image processing and computer vision has been slow due to its computational and storage complexity [3]. Algorithms to overcome the above drawbacks have been suggested by different authors. The HT has also been efficiently implemented on various types of multiprocessor systems. Most of the authors have used the straight line as an example of the pattern to be detected. They have either used the $m - c$ parametrization or the $\rho - \theta$ parametrization for the straight line. When $\rho - \theta$ parametrization is used, then the HT determines only the parameters ρ and θ. It does not provide any information about the length and position of the line segment.

In machine vision applications, the parameters as well as the length and position of the line are required for accurately locating the pattern. A line segment is completely defined by the normal parameters together with its position and length. The position of the line segment comprises of the co-ordinates of the endpoints of the line segment. Some authors have looked into this aspect and suggested various techniques to accomplish this objective. Some

of the methods suggested include the use of projection onto the image axes, surface fitting, and the use of the combinatorial Hough transform. The suggested methods are all highly compute bound and do not consider noise. It is worth mentioning here that the HT is usually applied to noisy images because of its robustness against noise. Therefore, any algorithm for determining the length and the endpoints of a line must take in to account the fact that the image will be noisy.

It is the objective of this thesis to determine the length of the line and as well as the co-ordinates of the endpoints of the line from the information contained in the Hough accumulator array. The amount of computation required in accumulating and analyzing the Hough accumulator array depends on the size of the array. Keeping this in view, it is also the objective of this thesis to determine the optimal size of the accumulator array. The optimal accumulator size will be investigated in relation to finding the complete line segment description from the accumulator array rather than just predicting the normal parameters of the line segment. The proposed approaches are based on exploiting the information contained in the spread of votes in the accumulator array. Algorithms for determination of complete line segment description in the presence of noise have also been investigated. All the approaches are highly efficient in terms of computing time.

1.5 Organization of the Thesis

The thesis is divided into six chapters. To familiarize the reader with this area of research, the Hough Transform and the various aspects associated with it are explained in chapters 2 & 3 respectively. Chapter 4 starts with a review of the state of the art in the area of complete line segment description along with their strengths and weaknesses. This is followed by the approaches proposed in this thesis. The optimal accumulator array size problem has been discussed while describing the proposed algorithms. Results obtained by using the proposed algorithms on synthetic images are presented in chapter 5. Wherever applicable the results are accompanied by observations and conclusions. An optimal accumulator array size has been recommended. The thesis concludes with a summary of the research carried out and suggestions for future work. There are two appendices in the thesis --- Appendix A gives the pseudo-code for the methodology employed to obtain the complete line segment description from the spread in the accumulator array, while Appendix B lists the FORTRAN code of the pseudo-code described in Appendix A.

2. THE HOUGH TRANSFORM

Historically, the Hough Transform (HT) has been the main means of detecting straight edges in image patterns [1]. It has also been applied to detect other geometrical shapes. The HT is used to estimate the parameters of a geometric shape in an image. The geometric shape is defined by a set of finite parameters , for example - a line by its slope m and intercept c, or by the length ρ and inclination θ of its normal, a circle by its radius r and centre (a,b) etc. The computational and space complexity of HT increases as the number of parameters increase.

In this chapter, a brief history of the HT is presented in sec 2.1 followed by a discussion about the various shape parametrizations suggested in literature for the line finding HT in sec 2.2.

2.1 Development of the HT

The HT was first introduced by Paul Hough in 1962 [44]. It was presented as a means to detect curves in bubble chamber photographs [3], and was brought to the attention of mainstream image processing community by Rosenfield [3]. In 1972, Duda and Hart [2] in an influential paper introduced a more pragmatic normal parametrization of straight lines. Several authors have used HT for detecting geometric shapes other than lines. Sklansky et-al [4]

explained how circular arcs can be detected and applied the technique to medical image processing. HT techniques for other geometric shapes such as parabolas and ellipses have been investigated by several authors - Sklansky and Wechsler [5], Tsuji and Matsumoto [6] and Tsukune and Goto [7]. Merlin and Farber [8] showed how the HT can be generalized to detect an arbitrary shape at a given orientation. Ballard [9 , 10] improved upon the work of Merlin and Farber in terms of efficiency.

2.2 Shape Parametrizations

The HT is capable of detecting any arbitrary shape which can be defined by a set of parameters. The HT converts a difficult global detection problem in image space into a more easily solved local peak detection problem in parameter space. The simplest geometric shape is a straight line. The HT for straight lines initially used the slope-intercept parametrization and later developments incorporated the normal parametrization of a straight line [2].

2.2.1 Slope - Intercept Parametrization

In this type of parametrization, a straight line is defined by its slope m with respect to x-axis and intercept c with the y-axis of the cartesian coordinate system. The line is defined by the equation

$$y = mx + c \qquad 2.1$$

The two parameters characterizing the line are m and c. Hence the parameter space is 2-dimensional. The parameter space is the $m - c$ plane defined by the equation

$$c = -mx_i + y_i \qquad 2.2$$

Eq 2.2 is the equation of a straight line corresponding to the feature point (x_i, y_i) as shown in fig 2.2.1.1. Another feature point (x_j, y_j) will also have a line in the parameter space intersecting with the line corresponding to (x_i, y_i) at (m', c'), m' and c' being the parameters of the line containing both (x_i, y_i) and (x_j, y_j) in the image space. All feature points in the image space will have corresponding lines in the $m - c$ parameter space intersecting at (m', c') [11].

The HT proceeds by dividing the parameter space into accumulator cells, which are set to zero initially. For a feature point (x_i, y_i) in the image space, the parameter m is assigned various sampled values on the $m -$ axis and the value of c is computed using eq 2.2. This value of c is quantized on the $c -$ axis in the $m - c$ parameter space for each sampled value of m. If a choice of m results in a solution to c in the parameter space, then the corresponding accumulator cell is incremented i.e;

$$A(m, c) = A(m, c) + 1 \qquad 2.3$$

where $A(m, c)$ is the cell in the accumulator array. This process is carried out

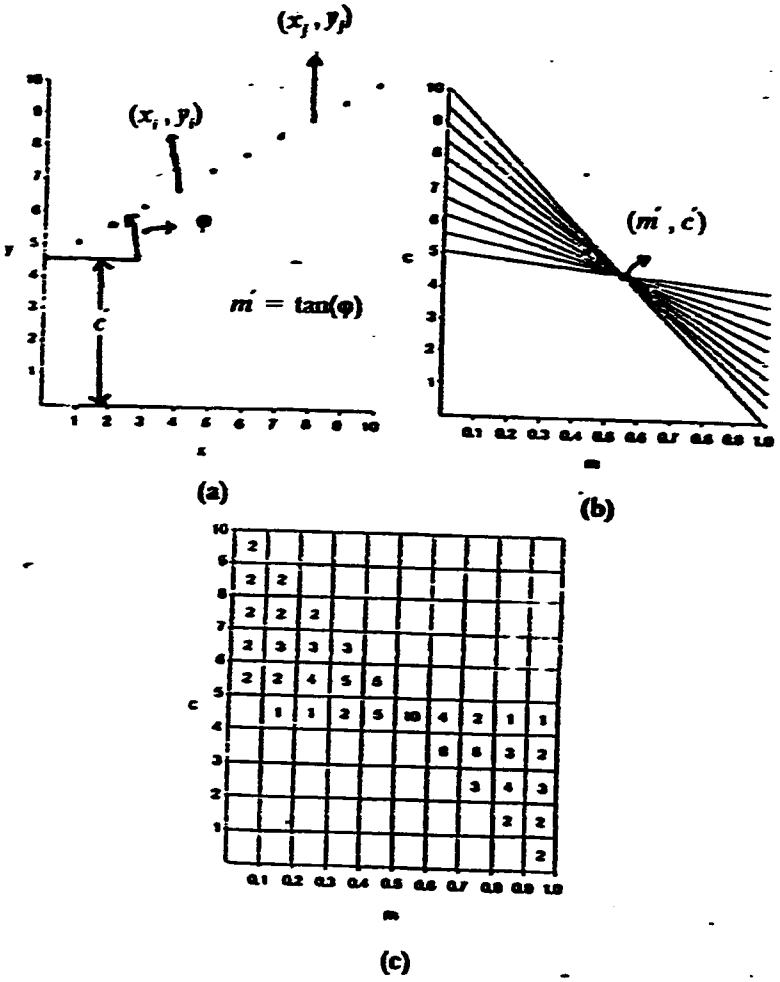


Fig 2.2.1.1 The basis of the Hough Transform for line detection (a) (x, y) image space (b) (m, c) parameter space (c) accumulator array around (m', c') in (b)

for each feature point in the image space. The parameters m' and c', of the line in the image are obtained by searching for peaks in the accumulator array. The co-ordinates of the cell having the maximum value or a value above a certain threshold are the parameters of the line in the image space. The accuracy of the parameters estimated depends on the resolution of m and c in the parameter space.

One of the drawbacks of the slope-intercept parametrization of straight line segments in the HT is that, both the slope and the intercept approach infinity as the line segment approaches a vertical position [11]. In other words, the parameters m and c are unbounded, and hence such a parametrization has a singularity for lines with large slopes i.e., $m \rightarrow \infty$ [3]. In 1972, Duda & Hart [2], suggested the normal parametrization of the line to overcome the problems arising due to the use of the slope-intercept parametrization.

2.2.2 Normal Parametrization

The parameters of a line in normal parametrization are the length ρ and orientation θ of the normal to the line from the image origin [2]. The equation of a straight line segment corresponding to this geometry is

$$\rho = x\cos\theta + y\sin\theta \qquad 2.4$$

The parameter space is the (ρ, θ) accumulator array. The normal

parameters are unique and every line in the image plane corresponds to a unique point in the $\rho - \theta$ plane [2]. The HT for lines using normal parameterization transforms a set of collinear feature points $\{(x_k, y_k), k = 0, 1, ..., n_f\}$, (where n_f is the number of feature points in the image plane) lying on the line in the image space to a set of sinusoidal curves in the parameter space defined by eq 2.4 as illustrated in fig 2.2.2.1. All these curves intersect at a point (ρ', θ') in the parameter space. This point gives the normal parameters of the line. Duda & Hart [2] have established a dual property of the point-to-curve transformation. The set of points $\{(\rho_1, \theta_1), (\rho_2, \theta_2), ...(\rho_k, \theta_k)\}$ in the $\rho - \theta$ plane all lying on the curve

$$\rho = x_o \cos\theta + y_o \sin\theta \qquad 2.5$$

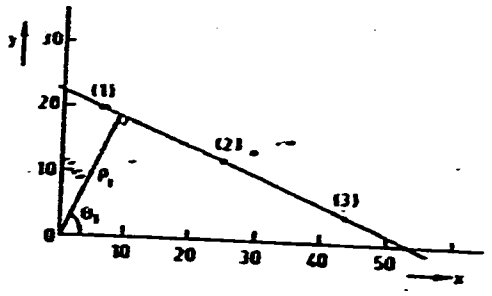
correspond to the lines in the image plane passing through the point (x_o, y_o). Some of the properties of the point-to-curve transformation are [2]:

1. A point in the image plane corresponds to a sinusoidal curve in the parameter space.

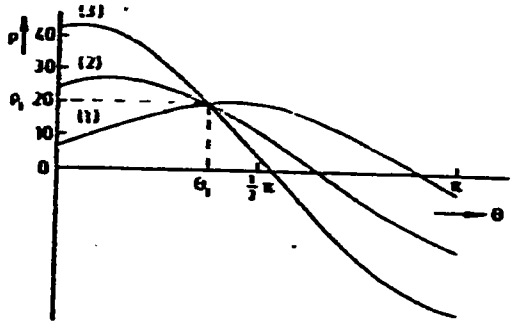
2. A point in the parameter space corresponds to a straight line in the image plane.

3. Points lying on the same straight line in the image plane correspond to curves through a common point in the parameter space.

4. Points lying on the same curve in the parameter space correspond to lines through the same point in the image plane.



(a)



(b)

Fig 2.2.2.1 (a) Line in normal form $\rho_1 = x\cos\theta_1 + y\sin\theta_1$ (b) sinusoidal curves corresponding to feature points on the line

In the HT for lines using normal parametrization, the parameter θ is sampled and the parameter ρ is quantized in the parameter or the transform space.

2.2.3 Other Parametrizations

Although normal parametrization described in the previous section is primarily used in computing the HT for straight lines, there have been several other parametrizations suggested in literature. Davies [12], suggested the *foot of the normal* parametrization. In this approach, a line segment is parametrized by the point of intersection of the normal from the origin with the line. If (x_o, y_o) is the foot of the normal then

$$\frac{g_y}{g_x} = \frac{y_o}{x_o} \qquad 2.6$$

$$(x - x_o)x_o + (y - y_o)y_o = 0 \qquad 2.7$$

where (g_x, g_y) is the gray-level gradient in the x & y directions for each edge pixel in the image. The gradient is obtained when the image is passed through edge detectors. Solving eq 2.6 and eq 2.7 for x_o and y_o gives

$$x_o = vg_x \qquad 2.8$$

$$y_o = vg_y \qquad 2.9$$

where,

$$r = \frac{xg_x + yg_y}{g_x^2 + g_y^2} \qquad 2.10$$

These expressions involve only additions, multiplications and divisions unlike the normal parametrization which involves trigonometric functions.

Wallace [13] introduced a new bounded parametrization for straight lines, and the modified HT is called the Muff transform. The line segment is parametrized by the two intersections or the projected intersection points with the perimeter of the image. These points are stored as distances s_1 and s_2 along the outer edges of the image. This representation is useful when using digital graphing techniques, and has the capability to provide uniform resolution for straight line segments.

The *feature point spread function* (feature psf) is the set of votes in the parameter space that result from a feature point in an image [14]. Though the normal parametrization of lines has been widely used in computing the HT for lines, but according to Davies [1] an open research issue is to find a parametrization that yields more nearly radially symmetric psf's for lines or show why such a parametrization does not exist.

2.3 Summary

In this chapter the principle of the HT and various shape parametrizations were discussed. The intricacies involved in the operation of the HT can be better understood with a insight of the various algorithms and performance aspects associated with the HT. These performance aspects such as the discretization errors and the quantization errors are explained in the next chapter followed by a brief description of the architectures for the HT and the applications of the HT.

3. VARIOUS ASPECTS OF THE HOUGH TRANSFORM

The HT for lines described in the previous chapter has many desirable features. The HT can be viewed as an evidence gathering procedure [3]. Each image point votes for all parameter combinations that could have produced it, if it were a part of the sought-after shape. The HT is closely related to template-matching techniques. One main difference is that the template-matching is carried out entirely in the image domain [3]. Template-matching involves generating from a basic template all other templates and determining how many of these template points match the image points. An inherent drawback of template-matching is the generation of redundant data, which is overcome in the HT. Some of the other sailent features of the HT are : [3]

1. The HT is highly robust to noise produced by poor image segmentation. Noise points do not contribute coherently to a single cell in the accumulator array, and therefore produce very low-level background counts in the array.

2. The HT is robust against discontinuities in the pattern.

3. The HT can simultaneously accumulate evidence for several occurances of a particular shape in an image.

4. The HT degrades gracefully to the phenomenon of occlusion which is a severe problem for other shape detection techniques.

5. Each image point is treated independently. This underlines the inherent parallelism offered by the HT and hence parallel processing of all points is possible.

6. The HT combines evidence independently, which helps in detecting partial or slightly deformed shapes.

The principal disadvantage of the HT is its large storage and computational requirements. The determination of q parameters of a pattern, each resolved into α intervals, requires an accumulator array of α^q elements [3]. This could be prohibitively large if either α or q is large. The dimensionality of the accumulator array increases with the number of parameters defining the geometric shape. The efficiency of the HT can be increased by devising methods which use small sized accumulator arrays or use prior information to reduce the range of parameters addressed [3].

In this chapter, the various aspects of the HT are discussed in detail. Sec 3.1 deals with the various algorithms suggested in literature for the HT. These include the *adaptive Hough transform* (AHT), the *fast Hough transform* (FHT) and several others. This discussion is followed by the analysis of the HT performance in sec 3.2. This analysis includes the discretization errors and the quantization errors occuring in the HT.

3.1 Algorithms for the HT

Various algorithms have been proposed in the literature, to overcome the drawbacks of storage and computational complexity of the HT. One common observation has been that high accumulator resolution is necessary only in places where high density of votes accumulate. A useful idea suggested by several authors is the iterative focusing of the HT to identify peaks with high counts. Initially the HT can be accumulated in a coarse uniform resolution parameter space. Regions with high counts can be investigated at higher resolutions. Consider searching for a peak in q − dimensional space resolved into α accumulator cells. Iterative use of much smaller accumulator array of size β requires $O(\log(\alpha)/\log(\beta))$ iterations to focus down to the same resolution. The computational saving of the method is proportional to the ratio of the number of cells in the two accumulators i.e., $(\alpha/\beta)^q$ times the number of iterations. Hence the computational saving is $O((\alpha/\beta)^q \times \log(\alpha)/\log(\beta))$ and the saving in the storage space is $O(\alpha/\beta)$ [3]. This provides high efficiency benefits in terms of both the amount of computation and storage space required.

Li et-al [15] have proposed a focusing algorithm called the *fast Hough transform* (FHT). The FHT uses multidimensional quadtree in conjunction with the HT which maps image points into hyperplanes. Computational gains of $O(10^3)$ for 2D lines and $O(10^{16})$ for 3D plane detection can be achieved, and easy implementation in VLSI is possible because of the regular structure of the FHT. The quadtree data structure is static and cannot optimally adapt to

situations which may require different parameter resolutions along different parameter axes [3].

Illingworth & Kittler [16] have developed an iterative coarse-to-fine search algorithm for detecting lines and circles in 2D and 3D parameter spaces. This algorithm is called the *adaptive Hough transform* (AHT). Parameter limits are defined independently for each parameter dimension, and hence appropriate precision for each parameter can be achieved. The benefit of using AHT in higher dimensions is greater than the two-dimensional case. The lack of regular structures like $k-D$ trees or pyramids distinguishes the AHT from other approaches to the HT [17]. After studying the properties of the AHT, Princen et-al [17] conclude that AHT is not applicable to a general class of object identification problems and has poor performance in complex scenes. The computational efficiency gains offered by the AHT hold only in the case when small number of objects have to be identified, since the computational cost of the AHT is directly proportional to the number of objects to be detected.

Brown et-al [18] discussed the use of addressable store i.e., hash table in software or a cache in hardware, in place of accumulator array to overcome the storage problems of the standard HT. This approach has a better performance than the standard HT when the environment in the image around the shape to be detected is good i.e., large segments with low noise are present. When conditions get worse, the average performance is approximately the same. Systematic distortion, occlusion and feature perturbation noise are severe problems for the cache Hough transform.

Several other approaches to HT accumulation have been suggested in literature. Princen et-al [19] discuss a hierarchical approach to line extraction based on the HT, which uses a pyramid structure for splitting the complete image into subimages. The algorithm proceeds bottom-up with short segments detected at the bottom level of the pyramid. The principle advantage is the absence of formation of insignificant peaks which occur in the standard HT. Wallace [13] suggested a modified Hough transform called the *Muff transform.* The main difference is that a different shape parametrization is used. Muff transform is better suited for computer graphics. Probability theory has also been used for developing efficient HT algorithms. Stephens [20] developed a strong relationship between the HT and the maximum likelihood method. Kriyati et-al [21] have discussed the *Probabilistic Hough Transform* (PHT). They use a small subset of edge points in an image selected at random as input for the HT. The performance of the PHT depends on the fraction of the edge points used and efficiency increases with an increase in the fraction of edge points used.

The original patent for the HT was not directly concerned with accuracy but simply the recognition of line segments. The HT was used to analyse the bubble chamber photographs of subatomic particle tracks, which contain large amount of noise. Since then HT has been implemented digitally and applied in machine vision and robot guidance. For this reason the issue of the accuracy the HT has become important.

3.2 Analysis of HT Performance

The initial work on the properties and performance of the HT was concerned with the effect of statistical measurement on the position and localization of the parameter peaks [3]. Shapiro & Iannino [22] and Sklansky [23] used geometric constructions for predicting the HT performance. Accuracy of parameter estimation and quantization errors were modelled using geometric constructions. Cohen & Toussiant [24] studied the density of counts produced in the parameter space by uniform distribution of image points. They used normalization of expected mean and standard deviation at each cell of the accumulator to improve the peak finding. Also, the effect of noise can be overcome by either an empirical background subtraction method or a non-linear quantization of the parameter ρ, such that each cell of the accumulator contains equal counts from random noise. Brown [14] studied the inherent noise associated with the HT. Analytic form of the peaks in the parameter space was developed.

Line segments are found in the Hough Transform by searching for peaks in the parameter space. A combination of factors influence the shape of these peaks. These factors include the discretization of the image plane, the width of the line, the quantization of the parameter space and the resolution of $\Delta\rho$ and $\Delta\theta$ in the parameter space. The resolutions $\Delta\rho$ and $\Delta\theta$ affect the memory space requirements and the computational complexity of the HT. [25, 26, 27] The most important point to be considered while developing how these factors

(particularly $\Delta\rho$ and $\Delta\theta$) should be chosen is that they all are dependent on each other. The choice of how large or small $\Delta\rho$ should be depends on that of $\Delta\theta$ and that of $\Delta\theta$ on the memory space requirements. If $\Delta\rho$ and $\Delta\theta$ are made too large, then memory space requirements and computational complexity will reduce at the expense of the accuracy of the estimated parameters. Unlike this, if $\Delta\rho$ and $\Delta\theta$ are made too small, then memory space requirements and computational complexity will increase, but the accuracy of estimated parameters is also increased. Thus a trade-off between accuracy desired and memory space requirements & computational complexity is necessary.

Another important factor on which the performance of HT is heavily dependent is the detection of peak. Accurate detection of the peak is dependent on the resolution of $\Delta\rho$ and $\Delta\theta$ [25]. The peak starts spreading in the ρ direction of the accumulator array, if $\Delta\rho$ is made too small. This affects the accuracy of estimated value of ρ of the line. If $\Delta\theta$ is made very small, then there will be horizontal elongation in the vicinity of the peak. In addition to this, the detection of peak becomes ambiguous and could mislead the peak detection algorithm.

Van Veen & Groen [25] studied the discretization errors in the HT for straight lines using (ρ, θ) parametrization. They also investigated the effect of quantization of the parameter space and the width of the line on the spreading of peak in the accumulator array. Niblack & Petkovic [27, 28] suggested a new method for HT which involves smoothing of the Hough array prior to peak searching and interpolation around the peak to increase the accuracy. The main

difference between this approach and the standard HT is in the accumulation of votes. In the standard HT, the value of ρ is computed using eq 2.4 for a sampled value of θ and quantized to the nearest value of ρ. The corresponding cell in the accumulator is incremented. Niblack & Petkovic linearly distribute this increment over a region in $A(\rho, \theta)$, the Hough accumulator array. Let (x_i, y_i) be a feature point and θ_k be a sampled value of θ. Then from eq 2.4

$$\rho_{exact} = x_i \cos\theta_k + y_i \sin\theta_k \qquad 3.1$$

If ρ_{low} and ρ_{high} are two consecutive quantized values of ρ on the $\rho-$ axes in the parameter space such that

$$\rho_{low} \leq \rho_{exact} < \rho_{high} \qquad 3.2$$

then increment $A(\rho_{low}, \theta_{peak})$ by an amount proportional to $(\rho_{high} - \rho_{exact})$ and increment $A(\rho_{high}, \theta_{peak})$ by an amount proportional to $(\rho_{exact} - \rho_{low})$. After finding the peak, interpolation is used to increase the efficiency.

3.2.1 Discretization Errors in the HT

The Hough Transform of a feature point (x_i, y_i) is performed by computing ρ from eq 2.4 for all n values of θ_k into which θ is sampled. The values of ρ are then quantized in m intervals of width Δρ. Strictly speaking θ is sampled and ρ is quantized, while computing the HT of pixels on a line

segment. [25] Each cell (ρ_i, θ_k) in the parameter space in which θ_k is sampled and ρ_i is quantized corresponds in the image plane to a bar-shaped window of infinite length and of width $\Delta\rho$ as shown in fig 3.2.1.1 [25, 26]. The detection of peaks when the image is divided using bar-shaped windows is dependent on ρ and θ of these windows. The number of feature points in the bar-shaped windows depends on the overlapping area between the image and the windows. This area is a function of the distance ρ of the bar from the origin and its inclination θ [25, 26].

Van Veen & Groen [25] have developed an expression for the spreading of the peak considering an infinitesimally thin line. According to them the number of cells n_ρ over which the peak spreads in the ρ – direction for a line of length l is given by

$$n_\rho = \left\lfloor \frac{l \sin(\Delta\theta/2)}{\Delta\rho} \right\rfloor + 2 \qquad 3.3$$

According to this expression, minimum peak spread of 2 occurs when $\Delta\rho$ is chosen such that

$$\Delta\rho \geq l_{max} \sin(\Delta\theta/2) \qquad 3.4$$

where l_{max} is the maximum length of the line segment. But the estimate of l_{max} is not known. An assumption is made that the length of the line is known, which is not true in real life images. One of the objectives of this thesis is to find l_{max}.

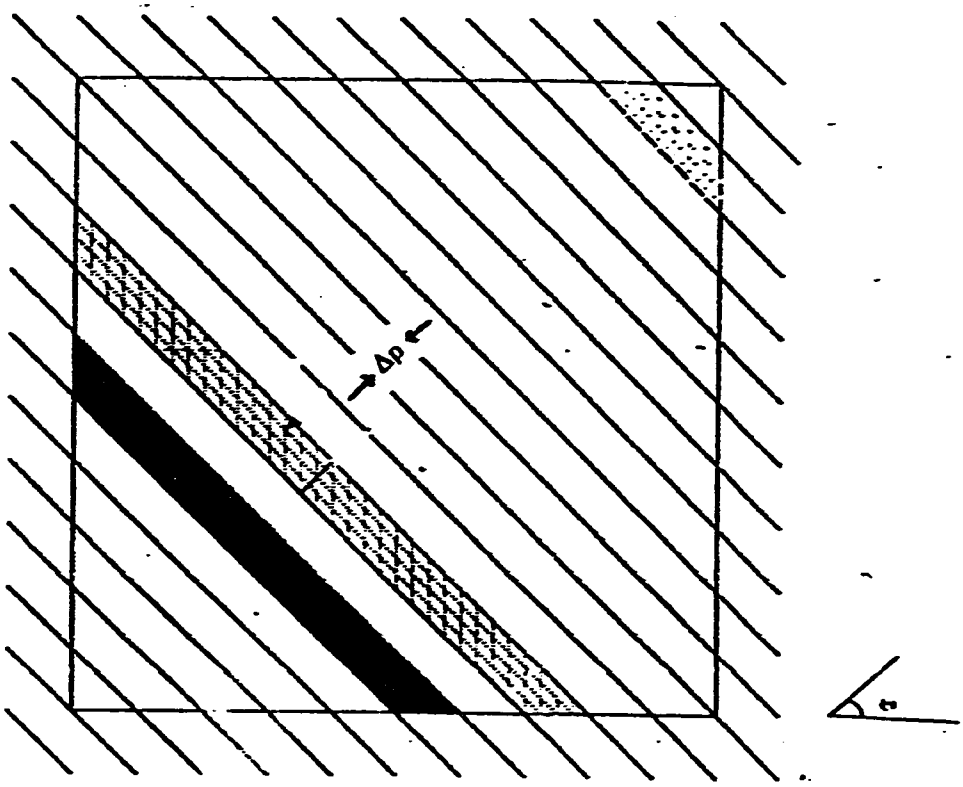


Fig 3.2.1.1 The overlapping area between the image space and an infinite window of width $\Delta\rho$.

The elongation of peak in the θ direction can be investigated by rotating a window corresponding to a (ρ , θ) cell around the centre (x_m ,y_m) of the line segment [25] as shown in fig 3.2.1.2. The window can be rotated over an angle α without crossing the line of length *l*. The extension of the flat upper part of the peak in the θ direction is

$$\alpha = 2\sin^{-1}(\Delta\rho/l) \qquad 3.5$$

If α is large, peak decays in both θ and ρ directions. Here again, an assumption is made that an estimate of length *l* is known. The smaller Δρ is chosen, the smaller is the elongation of peak in the θ direction. (from eq 3.5). But a decrease in Δρ increases the number of cells over which the peak spreads in the ρ direction (from eq 3.3). According to Van veen & Groen [25] neither oversampling nor undersampling occurs when

$$\Delta\rho \approx l\sin(\Delta\theta/2) \qquad 3.6$$

The problem of discretization errors in the Hough Transform is also referred to as the aliasing problem in literature. When votes corresponding to evidence of pixels align at more than one parallel line, aliasing is said to have occured. According to Srihari & Govindraju [26], this anamoly occurs when the sampled value of θ, denoted by θ_s becomes 45° and 135° during sampling of θ between 0° and 180° in all images tested by them. Every alternate ρ has the aliasing problem at these two orientations. They conclude that aliasing can occur at all those values of θ where tan(θ) is a rational number, because only in such cases,

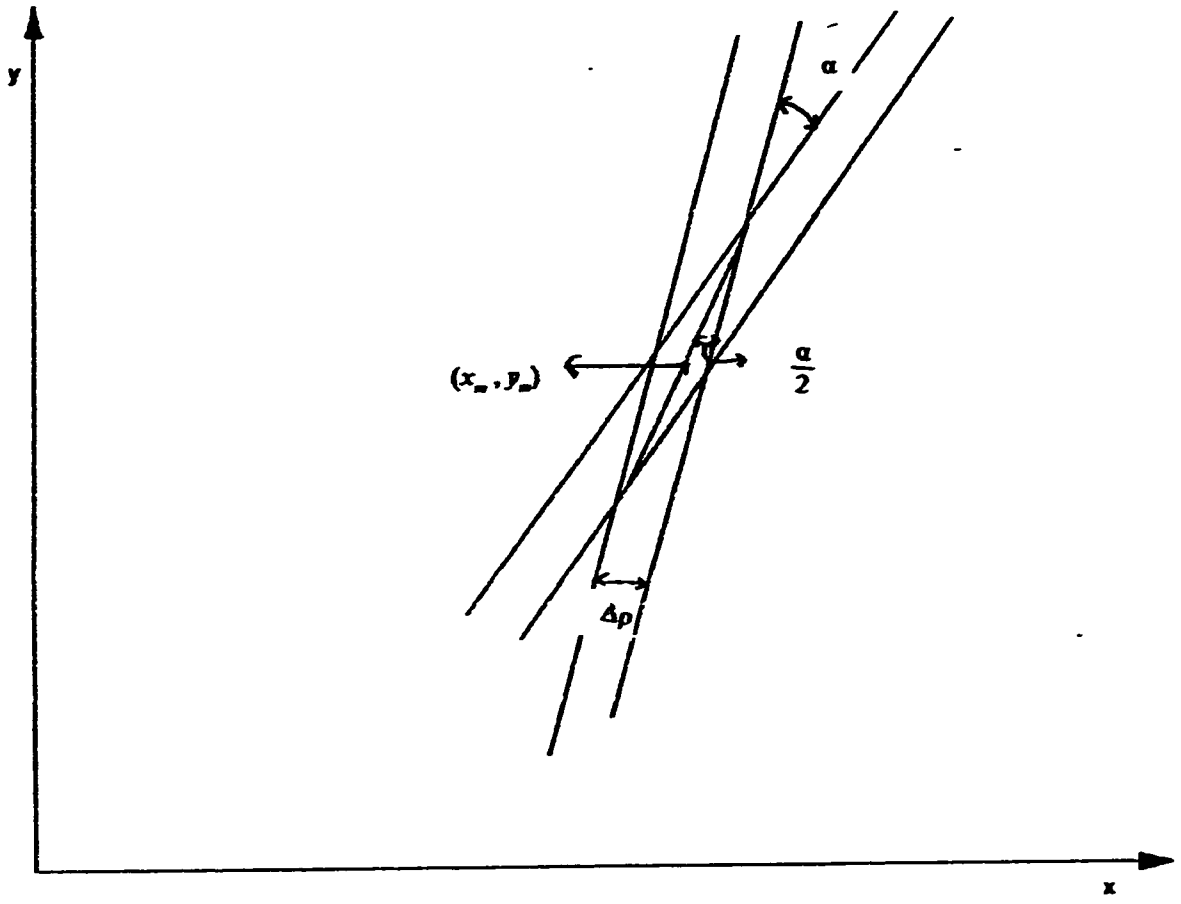


Fig 3.2.1.2 A bar corresponding to a $\rho - \theta$ cell can be rotated over an angle α without crossing the line segment of length l.

adjacent parallel lines will pass through pixels at regular intervals. But in real life applications aliasing can occur at other orientations as well. Srihari & Govindraju [26] sample θ at integral values only and use $\Delta\rho = 1$, hence the conclusion they arrived at holds. To overcome the aliasing problem, they suggest the width of the parallel windows or bars be made a function of the value of θ_s. Assuming h to be the grid distance, they suggest the following expressions

$$\Delta\rho(\theta_s) = |h\cos(\theta_s)| \quad -45^\circ \le \theta_s < 45^\circ,\ 135^\circ \le \theta_s < 225^\circ \qquad 3.7.a$$

$$\Delta\rho(\theta_s) = |h\sin(\theta_s)| \quad 45^\circ \le \theta_s < 135^\circ,\ 225^\circ \le \theta_s < 315^\circ \qquad 3.7.b$$

These expressions are similar to those suggested by Sandler et-al [42] and Sandler and Costa [29].

3.2.2 Quantization Errors in the HT

Points lying on a line having parameters ρ and θ need not be collinear after digitization [25] as shown in fig 3.2.2.1. Such type of lines are equivalent to lines having a certain width. When Hough Transform of such lines is computed, the accumulator array will also have a spread in $\rho-$ direction for θ_s close to θ_{actual}. Another important aspect of such lines is that the line is split up into smaller segments in the direction of one of the coordinate axes. When such a phenomenon occurs, there will be some feature points in the line having one of

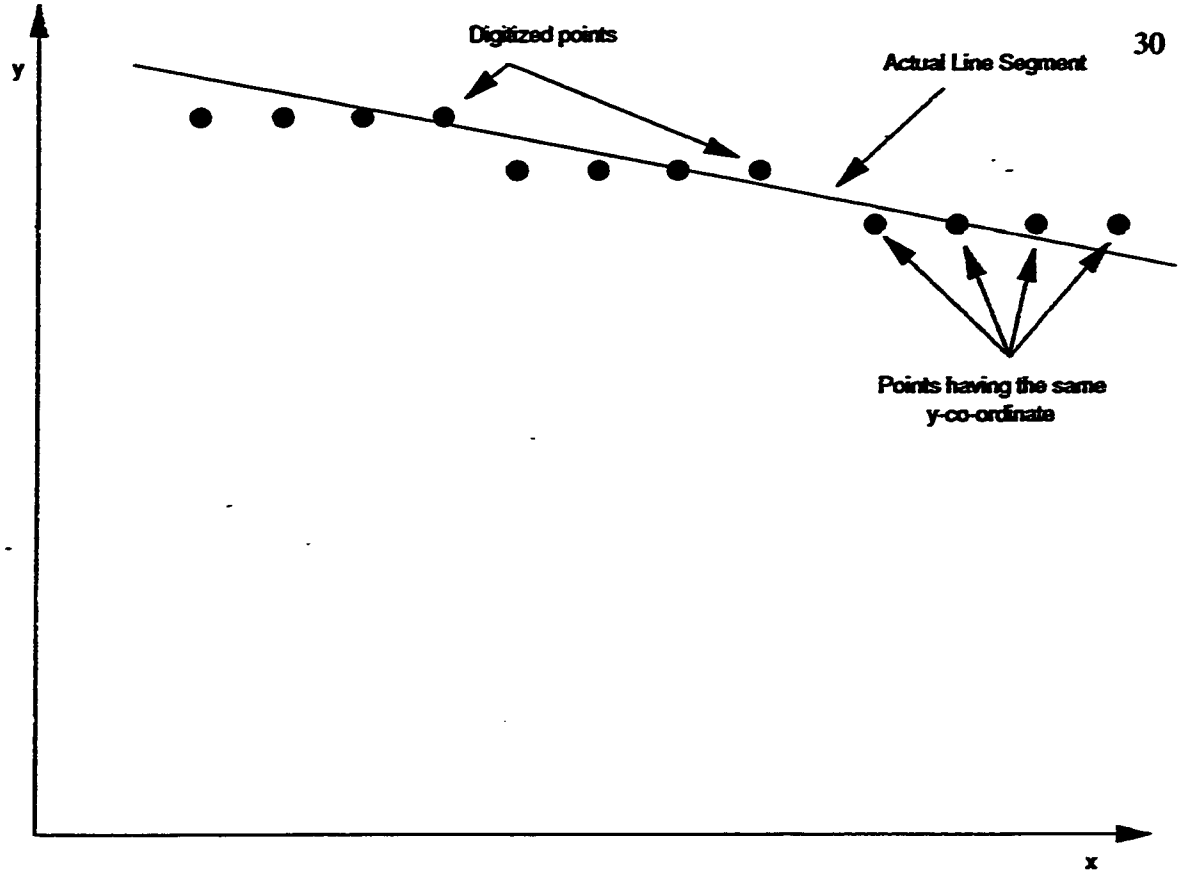


Fig 3.2.2.1 Points of a quantized straight line

the coordinates same for different values of the other coordinate. For a line having $\theta > 45^{\circ}$, then the feature points will have same y-coordinate for several values of the x-coordinate. While if $\theta < 45^{\circ}$, the feature points will have same x-coordinate for several values of the y-coordinate.

The apparent width of the line segment depends on θ_{actual} and the quantization scheme used. [25] According to Van veen & Groen [25], the maximum width of the line segment when Freeman's grid-intersection method having a grid constant h is used is $2d_m$ where

$$d_m = \frac{h}{2}\cos\theta \qquad 45^{\circ} \le \theta < 135^{\circ} \qquad 3.8.a$$

$$d_m = \frac{h}{2}\sin\theta \qquad -45^{\circ} \le \theta < 45^{\circ} \qquad 3.8.b$$

But this is not the apparent width of the line. The apparent width is b_a as shown in fig 3.2.2.2, which can be derived as

$$b_a = f_{same}\cos\theta \qquad 45^{\circ} \le \theta < 135^{\circ} \qquad 3.9.a$$

$$b_a = f_{same}\sin\theta \qquad -45^{\circ} \le \theta < 45^{\circ} \qquad 3.9.b$$

where f_{same} is the number of points having one of the coordinates equal, for example, $f_{same} = 4$ in fig 3.3.2.2. This can be computed by determining the slope $m_{predicted}$ of the line. An approximate value of the slope $m_{predicted}$ can be obtained from the predicted value of θ, denoted by $\theta_{predicted}$, of the line after one iteration

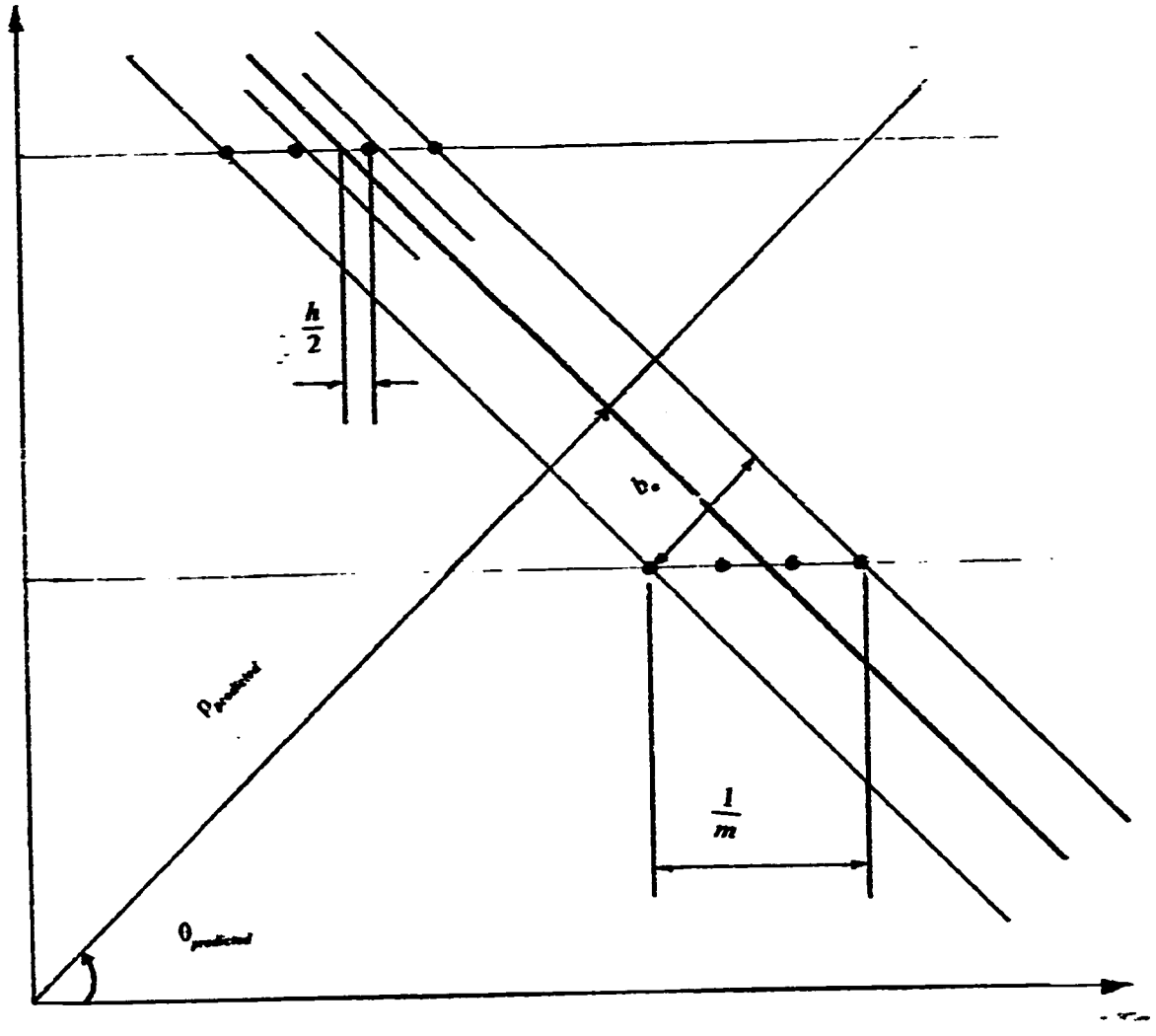


Fig 3.2.2.2 Apparent width of a thick line segment.

of HT. The slope of the line is given by

$$m_{predicted} = \frac{-\cos\theta_{predicted}}{\sin\theta_{predicted}} \qquad 3.10$$

where,

$$f_{same} = \frac{1}{m_{predicted}} \quad or \quad \frac{1}{m_{predicted}} + 1 \qquad 3.11$$

Another phenomenon of lines having apparent width is that, if $\Delta\rho$ is smaller than the grid constant h, then completely or almost empty windows pass through the line resulting in regular structure of no votes or low votes in the accumulator array in the vicinity of the peak.

Due to the compute-intensive nature of the HT, several architectures have been investigated in the literature. It has been found that the HT lends itself to parallel processing. Several architectures for the HT are discussed in the next section.

3.3 Architectures for the HT

One of the main charecterstics of the HT is that it consists of a series of fairly simple calculations which are carried out independently on every feature point in the image space. Most of the architectures suggested for implementing the HT have been restricted to the $\rho - \theta$ line finding HT [3]. All these

approaches try to exploit the inherent parallelism offered by the HT. Hanahara et-al [30] have implemented the $\rho - \theta$ line finding HT which pipelines the ρ intersection calculation and the accumulator incrementation steps. MC68000 was used as the main processor. Rhodes et-al [31] used the techniques of restructable VLSI (RVLSI) to produce a HT processor from a standard wafer containing many sum and multiply cells. This processor runs at frame rates.

The mapping of the HT onto Single Instruction Multiple Data (SIMD) architectures was first suggested by both Silberberg [32] and Li [33]. Silberberg suggested distributing both an image feature and a cell of parameter space among all the processing elements (PE's). The SIMD architecture used was called Geometric Arithmetic Parallel Processor (GAPP). Li considered two schemes for running the FHT suggested by Li et-al [15]. In the first scheme each PE is assigned an image feature while the coordinates of a parameter cell are broadcast simultaneously to every PE by a central controller. In the second approach, each PE is assigned a volume of parameter space and the image features are broadcast. Little et-al [34] describe an implementation of the HT on an architecture called the *Connection Machine.* This is based on a 12-dimensional hypercube in which each processing element can be reached from any other by traversing at most 12 edges of the cube. Guerra & Hambrusch [35] and Rosenfield [36] have proposed parallel algorithms for line detection on mesh connected SIMD architectures. Both utilize the Massively Parallel Processor (MPP).

Fisher & Hingham [37] compute the HT on a machine called Scan Line

Array Processor (SLAP). This is a linear array of SIMD PE's in which each PE is assigned to a single column of the image and the array moves over the image to process all pixels of a row concurrently. Bowyer et-al [38] suggest a Multiple Instruction Multiple Data (MIMD) hypercube architecture for the HT. They use a 64 processor NCUBE. Brown [18] uses content addressable memory architecture to implement the HT. Instead of the accumulator array, a hash table in software or a cache in hardware is used. Recently Ben-Tzvi [39] suggested a synchronous multiprocessor implementation of the HT. In this scheme both the set of image features and the parameter space are distributed on a real-time MIMD distributed memory architecture. This algorithm has been implemented on Overlapped Shared Memory Multiprocessor Architecture (OSMMA) system.

3.4 Applications of the HT

The HT has proved to be a valuable shape detection tool in a wide range of machine vision problems. One of the major reasons for this is that straight lines and other simple geometric shapes are a common occurence in most of the natural and man made scenes [3]. HT has also been applied in the field of seismography for detecting linear as well as hyperbolic features. It is particularly well suited technique for seismic analysis because of the cluttered and fragmented nature of the seismic data [3]. Text analysis has also been performed using HT [26]. It has been experimented for counting the ridges in an automatic

fingerprint analysis system [3]. Skingley & Rye [40] have used the HT to detect faint lines in synthetic aperture radar (SAR) images. HT has also been applied to detect cleavage cracks in minerals and rocks [3].

Apart from the above mentioned applications, HT has been widely used to detect geometric shapes such as circles, polygons, parabolas, hypercubes etc.

3.5 Summary

Different algorithms for the HT and various performance aspects of the HT were explained in this chapter. With this background, the different approaches for obtaining the complete line segment description available in literature (which are limited in number) are discussed in the next chapter. The approaches are briefly explained and their drawbacks are pointed out. The proposed approach and its modifications are then explained in detail.

4. COMPLETE LINE SEGMENT DESCRIPTION

The complete line segment description includes the coordinates of the endpoints (or extremities) of a line in the image plane, together with its length l, the length and angle of the normal from the origin, ρ and θ respectively. Several authors have done work in this area of Hough Transform(HT). The common approach of all these authors is to use the information accumulated in the accumulator array indirectly. In the subsequent sections the work of these authors is briefly discussed and their drawbacks are pointed out, followed by the proposed methods for thin lines and thick lines.

4.1 Determination of extremities by projection

Yamato et-al [41], perform the line segment detection by using feedback from the parameter space to the image plane. The Hough Transform is used to obtain approximate positions of the segments in the image plane and then extensive analysis is performed in the image plane. The detection of short segments is accomplished by eliminating Hough curves corresponding to the longer segments already detected. The algorithm proceeds by computing the parameters of the *feedback straight line* by using the conventional HT. This line is the *first likely line*, l_1 , given by

$$\rho_j = x\cos\theta_i + y\sin\theta_i \qquad 4.1$$

The second step in the algorithm is to estimate the number of feature points (black pixels), in l_1. The estimated number is assumed to be N_1 and is given by

$$N_1 = E + F + G \qquad 4.2$$

where E is the number of votes in the cell $(\rho_{peak},\theta_{peak})$ of the accumulator array, F and G are the number of votes in the cells $(\rho_{peak} + 1,\theta_{peak})$ and $(\rho_{peak} - 1,\theta_{peak})$ respectively.

The *second likely straight line* l_2 , is obtained by moving a square window of size $2M$ having the diagonal points as (x_l+M, y_l+M) and (x_l-M, y_l-M), where (x_l, y_l) is a point on l_1. This is given by

$$y_l=[(\rho_j - x_l\cos\theta_j)/\sin\theta_j] \qquad 4.3.a$$

where

$$x_l=2Mn, \quad (n = 1 \text{ to } [255/2M]) \qquad 4.3.b$$

Initially the operation is started with $M = 1$. Let the number of pixels obtained be n_1, which are represented by $\{(x_s, y_s), s = 1, 2, \ldots\ldots n_1\}$. If $n_1 < 0.7N_1$, then M is increased by 1. Once $n_1 > 0.7N_1$, the operation is stopped and that value of M is M_1. A straight line which approximates these pixels is obtained in the slope-intercept form as

$$y = mx + c \quad 45^\circ \le \theta_i < 135^\circ, \quad 225^\circ \le \theta_i < 315^\circ \qquad 4.4.a$$

$$x = my + c \quad -45^\circ < \theta_i < 45^\circ, \quad 135^\circ < \theta_i < 225^\circ \qquad 4.4.b$$

The line l_2 is computed by minimising $\sum_{s=1}^{n_1}\varepsilon_s^2$, where ε_s is given by

$$\varepsilon_s = y_s - mx_s - c \quad (s = 1, 2, \ldots\ldots n_1) \tag{4.5}$$

The normal parameters of l_2 are ρ_2 and θ_2.

The second line l_2 is tested for closeness to the actual line by estimating the number of black pixels n_2 on l_2. This is done by moving a window of size $2M_1$ on l_2. If $n_2 < 0.95N_1$, then n_2 is calculated with a larger window as was done for finding l_1. The operation of increasing M_1 and calculating n_2 is repeated until $n_2 > 0.95N_1$ or a maximum of 5 times, after which the m and c having the least error are used. The straight line finally obtained is l_t, which is given by

$$\rho_t = x\cos\theta_t + y\sin\theta_t \tag{4.6}$$

This algorithm detects segments in descending order of their lengths by subtracting Hough curves of already detected line segments. The endpoints of a line segment are computed by projecting onto x-axis (y-axis if $-45^\circ \le \theta_t < 45^\circ$ or $135^\circ \le \theta_t < 225^\circ$), the black pixels of the nearest line l_t. This projection is mapped into an array I, whose elements are 1 for the black pixels and 0 for the background. The array I is scanned from beginning to end. If an element of the array is 1, then it is taken as the *likely start* of the line segment l_t. The *likely end* is the element in the array having a 1, such that, all the elements in the array between these the above two elements contain a 1. The above two elements in the array are taken as the likely x-co-ordinates of the endpoints of the line

segment l_i, (y-co-ordinates if $-45^{\circ} \le \theta_i < 45^{\circ}$ or $135^{\circ} \le \theta_i < 225^{\circ}$). If there are multiple set of endpoints (which is always possible), then the set containing the maximum number of 1's is used.

This algorithm has the following drawbacks:

1. The accumulation of votes is carried out twice, first for accumulating the array and second during elimination of Hough curves of already detected line segments. In other words this algorithm uses *positive* and *negative* accumulation.
2. The number of feature points lying along the line segment is assumed to be N_1 i.e., the total of the number of votes in 3 adjacent cells in the bar with the peak in the middle. But the number of feature points will be the summation of votes in all those cells over which the peak spreads and not just 3 bars. The spread is given by eq 3.3.
3. The algorithm does not develop analytically the method to select the window width M.
4. Multiple endpoints could be detected for the same line.
5. The authors do not discuss whether the algorithm detects the endpoints of a thick line. In real-time applications thick lines having certain width are inevitably present in the image plane.
6. Detection of endpoints in presence of noise is not discussed.

4.2 Determination of extremities using the Combinatorial Hough Transform

Sandler et-al [42], proposed a new algorithm for the Hough Transform called *the Combinatorial Hough Transform* or CHT. The authors also suggest an improved parametrization technique called the *Improved Parameter Space Normalization* or IPSN. The principle behind IPSN is to vary the value of $\Delta\rho$ as a function of θ. Thus the resolution $\Delta\rho(k)$ at the k^{th} value of θ is defined as

$$\Delta\rho(k) = (\rho_{max}(k) - \rho_{min}(k))/(\rho_{asize} - 1) \quad , \quad 0 \le k \le \theta_{asize} - 1 \qquad 4.7$$

with

$$\rho_{max}(k) = N\sin(k\,\Delta\theta) + N\cos(k\,\Delta\theta) \qquad if \quad \cos(k\,\Delta\theta) \ge 0 \qquad 4.8.a$$

$$\rho_{max}(k) = N\sin(k\,\Delta\theta) \qquad if \quad \cos(k\,\Delta\theta) < 0 \qquad 4.8.b$$

$$\rho_{min}(k) = 0 \qquad if \quad \cos(k\,\Delta\theta) \ge 0 \qquad 4.8.c$$

$$\rho_{min}(k) = N\cos(k\,\Delta\theta) \qquad if \quad \cos(k\,\Delta\theta) < 0 \qquad 4.8.d$$

where N is the image size. In the conventional parametrization $\Delta\rho$, ρ_{max} and ρ_{min} are not functions of discrete value of θ and are given by

$$\rho_{max} = \sqrt{2}(\rho_{asize} - 1) \qquad 4.9.a$$

$$\rho_{min} = -N + 1 \qquad 4.9.b$$

$$\Delta\rho = (\rho_{max} - \rho_{min})/(\rho_{asize} - 1) \qquad 4.9.c$$

Srihari & Govindraju [26] also use a parametrization in which the resolution $\Delta\rho$ is a function of discrete value of θ.

In their approach, Sandler et-al use the contribution of each pixel for determining the endpoints. The HT is recalculated for only those angles and normal distances that correspond to peaks in the accumulator array. The algorithm uses a technique in which histograms corresponding to pixels which contribute to the peak are updated. A *connectivity analysis* is performed through these histograms to determine the extremities of the line segments. The algorithm can be outlined as follows:

For each peak (ρ_i', θ_i') in the accumulator array, whose count is larger than a given threshold τ_c (which is chosen as a function of shortest expected line segment), do the following

1. For each pixel (x_j, y_j) in the image, check if the inequality given by eq 4.10 is satisfied for (ρ_i', θ_i')

$$|\rho_i' - x_j \cos\theta_i' - y_j \sin\theta_i'| < \Delta\rho \qquad 4.10$$

   If the condition is satisfied include it in the corresponding histogram.

2. Perform a connectivity analysis through these histograms to determine the extremities of the straight line. Remove the detected line from the image plane.

The connectivity analysis involves searching for runs of more than $(\tau_c - 1)$ consecutive non-zero values. For line segments with discontinuities due to noise and occlusions, a maximum threshold τ_g for the gap length is also considered. This gap is chosen as the largest gap expected in a line segment.

Each histogram is a projection of the line onto the x or y axis. For getting accurate results, the following heuristic can be applied depending on the predicted value of θ from the accumulator array using CHT i.e., $\theta^{CHT}_{predicted}$. Use the projection of the histogram onto y-axis if $0^\circ \le \theta^{CHT}_{predicted} \le 45^\circ$ or $135^\circ \le \theta^{CHT}_{predicted} \le 180^\circ$ and use the projection onto x-axis if $45^\circ < \theta^{CHT}_{predicted} < 180^\circ$. If the y-co-ordinates of the endpoints are determined from the projection of the histogram, then eq 4.11 can be used for computing the x-co-ordinates. On the other hand if the x-co-ordinates of the endpoints are determined from the projection of the histogram, then eq 4.12 can be used for computing the y-co-ordinates. In eq 4.11 and eq 4.12, $\rho^{CHT}_{predicted}$ is the predicted value of ρ using CHT.

$$x = (\rho^{CHT}_{predicted} - y \sin(\theta^{CHT}_{predicted})) / \cos(\theta^{CHT}_{predicted}) \qquad 4.11$$

$$y = (\rho^{CHT}_{predicted} - x \cos(\theta^{CHT}_{predicted})) / \sin(\theta^{CHT}_{predicted}) \qquad 4.12$$

Several drawbacks of the above procedure for detecting extremities of line segments are

1. The values of two thresholds used i.e., τ_c and τ_g are not based on any formulations but they are chosen as functions of expected length of shortest line segment and maximum gap in a line segment respectively. But in practical applications, when an image is received for processing, use of expected values of τ_c and τ_g could lead to computation of rather inaccurate extremities.

2. The method may truncate longer line segments. These may be described by two or more shorter ones.

3. The performance of the algorithm for detecting thick lines which are inevitably present in real life images is not presented.

4. The performance of the algorithm in presence of noise or when several lines are present is not considered.

4.3 Determination of the extremities using Surface Fitting

Niblack & Truong [43], describe a model in which the endpoints of each segment and number of pixels along the segment are computed assuming a noise-like model. This approach is based on fitting a surface locally to the

Hough accumulator array, where the variables of the fit are the parameters being searched for. The accuracy of computed values of ρ and θ is improved by using two dimensional interpolation on the contents of the accumulator array. The feature points in the image around the line segment are assumed to have a gaussian distribution with mean zero and standard deviation σ_o in the direction orthogonal to the line segment. The average number of pixels along the line segment is μ_o. The centre of the line segment is (x_o,y_o) , length is L_o and normal parameters are ρ and θ. A is the actual accumulator array and $\hat{A}$ is the array obtained by taking noise into account with two dimensional interpolation.

The authors perform a non-linear least square minimization of $\Sigma(\hat{A}-A)^2$ in the vicinity of the peak. An initial guess of ρ_o and θ_o is obtained from a coarse estimation. The peak is detected using a window after smoothing the accumulator array in the ρ direction. The endpoints are computed by examining the columns of $A(\rho_o,\theta_o) \pm T_\theta$ based on the value of θ_o. In these columns minimum and maximum ρ values, corresponding to bars enclosing the same number of points as in the peak are determined. The two sets of ρ and θ for each endpoint are solved and hence an estimate of length L_o is obtained. Accuracy is further improved by using iterations. These iterations are stopped if $\Sigma(\hat{A}-A)^2$ decreases by less than 0.01 on two successive iterations. Typically six iterations are required. The size of the window used for smoothing the accumulator array affects the accuracy of the estimated parameters. The main drawbacks of this method are:

1. Only one segment at a given ρ and θ can be detected. Separate colinear line segments cannot be detected.

2. The accuracy of the estimated parameters for a thick line segment having certain width is not evaluated.

4.4 The Proposed Approach

In this approach, the spread of votes in the Hough accumulator array is utilized to compute the length of the line segment as well as its endpoints. The basis of the approach is the fact that each cell in the accumulator array corresponds to a bar shaped window of infinite length and width Δρ as shown in fig 3.2.1.1 [25 ,26]. In the subsequent discussion the following notation will be used. $A = \{a_{i,j},\ 0 \le i \le \rho_{asize} - 1,\ 0 \le j \le \theta_{asize} - 1\}$ is the Hough accumulator array, with i being the index in the ρ − direction and j being the index in the θ − direction in the parameter space. $b_{i,j}$ is a bar in the image plane having parameters corresponding to $a_{i,j}$. C_k is the k^{th} column in the Hough accumulator array as shown in fig 4.4.1. The spread in C_k in the ρ − direction will be denoted by n_ρ^k. $(\rho_{actual}, \theta_{actual})$ are the actual parameters of the line segment L. $(\rho_{predicted}, \theta_{predicted})$ are the predicted parameters of the line segment L, obtained from the peak in A. (ρ, θ) of a line can also be determined if the co-ordinates of the extremities of the line are known. Let $(\rho_{computed}, \theta_{computed})$ be the parameters of

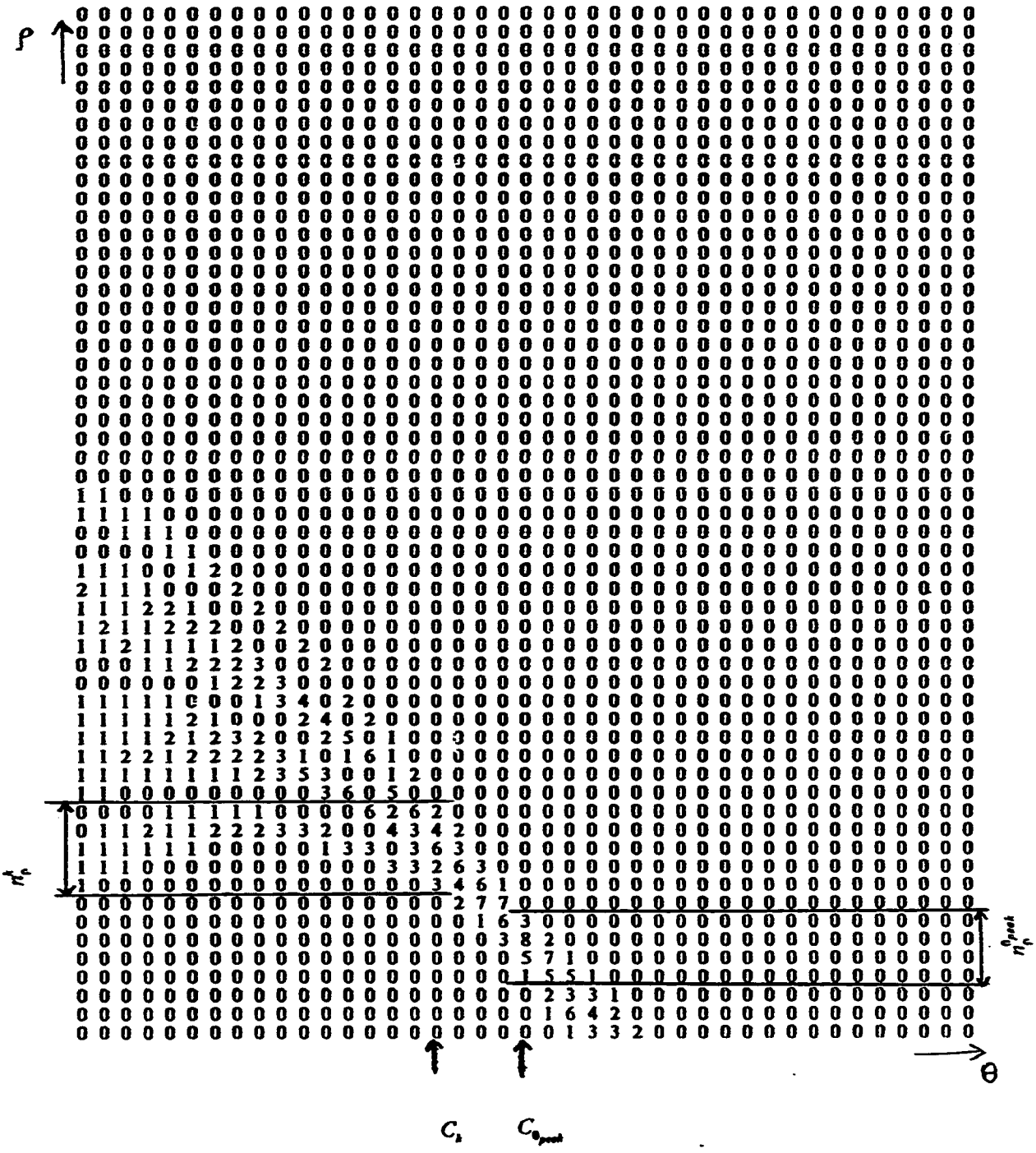


Fig 4.4.1 Part of a typical accumulator array illustrating $C_{\theta_{peak}}$, $n_\rho^{\theta_{peak}}$, C_k and n_ρ^k

the line segment L computed from the extremities.

4.4.1 Computation of length from the spread in the accumulator array

For a infinitesimally thin line segment L of length l_{actual} and parameters $(\rho_{actual}, \theta_{actual})$, the angle θ_{actual} in general will not be exactly equal to any of the sampled value $\theta_k, 0 \le k \le \theta_{axis} - 1$. Hence, the line segment may cross several consecutive bars having normals making an angle θ_k, with the positive x-axis, resulting in a spread of the peak in the ρ – direction. The number of cells in the accumulator array over which the peak spreads for a given $\Delta\rho$ and $\Delta\theta$ is given by eq 3.3. This expression for the spread can be simplified as

$$n_\rho^{\theta_{peak}} \approx \frac{l_{actual} \sin(\Delta\theta/2)}{\Delta\rho} + 1 \qquad 4.13$$

This can be best explained through an example. Assume that the expression $\frac{l_{actual} \sin(\Delta\theta/2)}{\Delta\rho}$ has a value of 1.9 . From eq 3.3 the maximum value of the spread $n_\rho^{\theta_{peak}}$ is 3. The probability that the spread will be in 3 bars for this value of the expression is greater than that it will spread in 2 bars. And from eq 4.13 the value of spread is 2.9 which is close to 3. Hence eq 4.13 can be used as an approximation for eq 3.3.

The maximum angle between the line segment L and $b_{\rho_{peak},\theta_{peak}}$ is $\frac{\Delta\theta}{2}$ as

shown in fig 4.4.1.1 . Let $d_{j,k}$ be the number columns between C_j and C_k, i.e., $d_{j,k} = |j - k|$. For all $b_{i,j}$, the maximum angle between the line segment L and $b_{i,j}$ is $\frac{\Delta\theta}{2} + d_{j,\theta_{peak}}\Delta\theta$. The spread in the ρ − direction in a column C_j is given by

$$n_\rho^j = \frac{l_{actual}\sin(\Delta\theta/2 + d_{j,\theta_{peak}}\Delta\theta)}{\Delta\rho} + 1 \qquad 4.14$$

But the value of l_{actual} in this expression is not known. Except l_{actual} all other terms appearing in eq 4.14 are known. Hence eq 4.14 can be rearranged to yield an expression for l as

$$l = \frac{(n_\rho^j - 1)\Delta\rho}{\sin(\Delta\theta/2 + d_{j,\theta_{peak}}\Delta\theta)} \qquad 4.15$$

Since our aim is to make $n_\rho^{\theta_{peak}} = 1$, we cannot determine the length of the line using $C_{\theta_{peak}}$. Hence the length of the line can be computed from the spread in C_j where $j \neq \theta_{peak}$. The accuracy of the computed length can be improved by computing l at C_js and averaging them. The averaged length is given by

$$l_{ave} = \frac{1}{2M}\sum_{j=-M, j\neq 0}^{M}\frac{(n_\rho^j - 1)\Delta\rho}{\sin(\Delta\theta/2 + d_{j,\theta_{peak}}\Delta\theta)} \qquad 4.16$$

for different values of M. The spread in $C_{\theta_{peak}}$ is one and hence this column is not taken in the summation. The value of M depends on the accuracy desired.

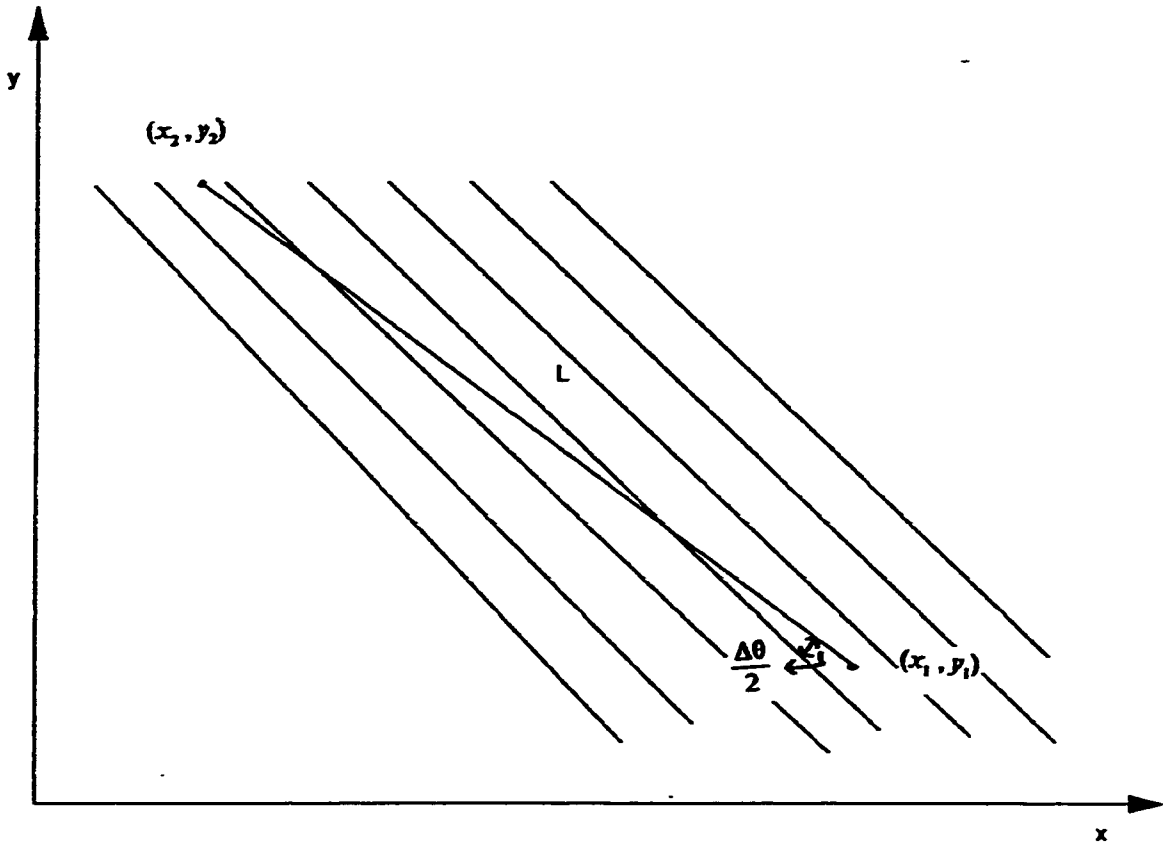


Fig 4.4.1.1 Maximum angle between the line segment L and bars in $C_{b_{peak}}$.

From simulation results on digitized images typical values of M between 1 to 5 gave accurate results. The accuracy of computed length also depends on the values of $\Delta\rho$ and $\Delta\theta$. Large values of $\Delta\rho$ and $\Delta\theta$ will result in a loss of accuracy, but reduce the computational complexity and the memory space requirements of the HT. Small values of $\Delta\theta$ will increase the computational complexity and the memory space requirements while small values of $\Delta\rho$ will lead to an increase in the latter. Some of the simulation results are shown in Tables 5.1.1 and 5.1.2 which are discussed in detail in sec 5.1. Another effect of small values of $\Delta\rho$ is the spreading of votes in $C_{\theta_{peak}}$. This can result in misleading the peak detection algorithm, and peak detection could itself become ambiguous, which will result in a failure to detect $\theta_{predicted}$ and $\rho_{predicted}$ properly. Using this approach only the length of the line segment can be determined, while our objective is to obtain the complete line segment description of the HT. Several approaches to compute the length and extremities of the line segment are discussed in the subsequent sections.

4.4.2 Length and Extremities of a Thin Line Segment

Digitized line segments are classified as thin or thick based on the angle of the normal to the line from the image origin. In the first quadrant, a *thin* line segment is one having its normal inclined at exactly 45°. Lines having their normal inclined at angles other than $\theta = 45^{\circ}$ are all *thick* line segments. In

thick line segments several consecutive feature points will have one of the co-ordinates equal for varying value of the other co-ordinate. Examples of thick lines for $\theta > 45^{\circ}$ and $\theta < 45^{\circ}$ are illustrated in fig 4.4.2.1 and fig 4.4.2.2 .

The endpoints of a line segment can be determined by using the spread in the accumulator array as shown in fig 4.4.2.3. In any column C_k, the cell corresponding to the first non-zero count detected while scanning from $i = 0$ to $\rho_{asize} - 1$ in the ρ – direction is called the first non-zero entry of that column. Similarly for any column C_k, the cell corresponding to the last non-zero count detected while scanning from $i = 0$ to $\rho_{asize} - 1$ in the ρ – direction is called the last non-zero entry of that column. Let ρ_1 be the length of the normal to the bar corresponding to the first non-zero entry in any C_k. ρ_1 is then given by

$$\rho_1 = \rho_{min} + \eta_{fn}^{k} \Delta\rho \qquad 4.17$$

where η_{fn}^{k} is the index in the ρ – direction, of the first non-zero entry in C_k, as shown in fig 4.4.2. The first non-zero entry in any C_k corresponds to the endpoint (x_2, y_2) which lies in $b_{\eta_{fn}^{k}, k}$ as shown in fig 4.4.2.3 . Similarly ρ_2 is the length of the normal to the bar corresponding to the last non-zero entry and is given by

$$\rho_2 = \rho_{min} + \eta_{ln}^{k} \Delta\rho \qquad 4.18$$

where η_{ln}^{k} is the index in the ρ – direction, of the last non-zero entry in C_k, as shown in fig 4.4.2. The last non-zero entry in any C_k corresponds to the

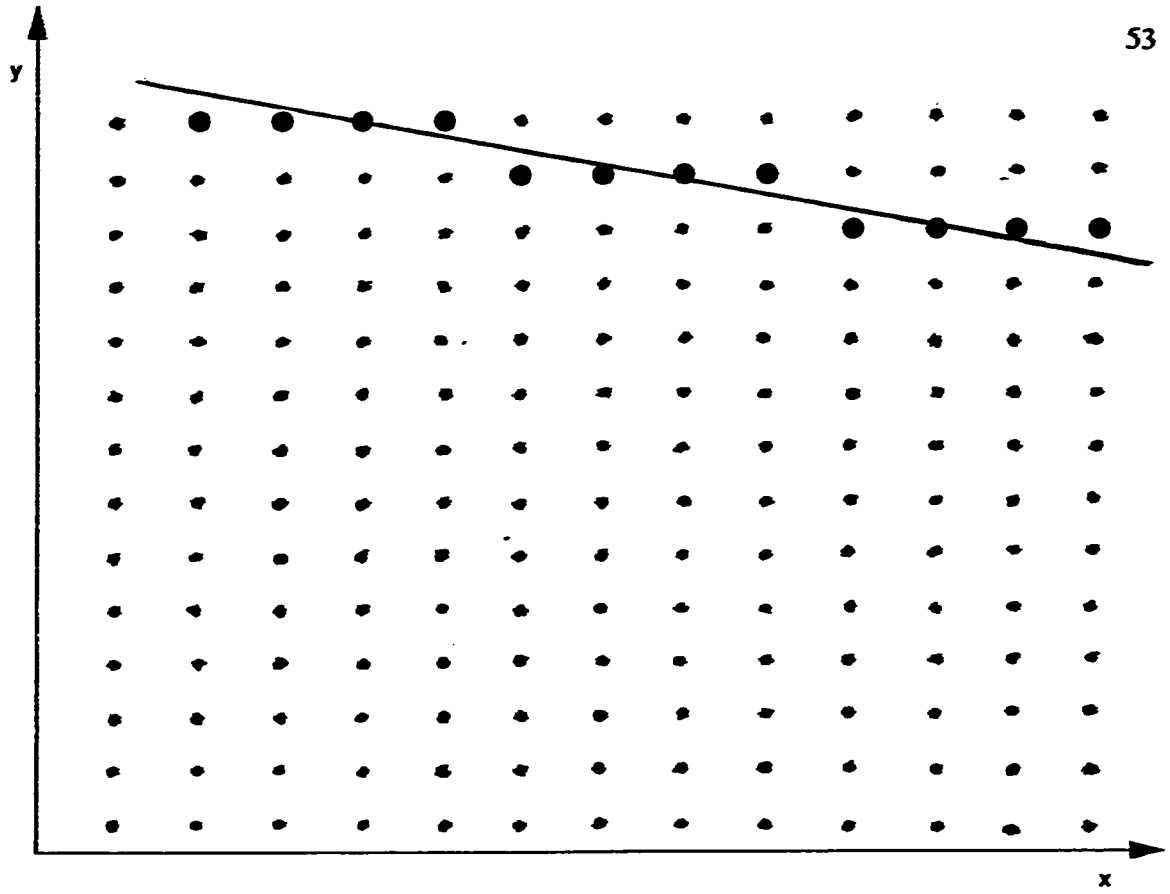


Fig 4.4.2.1 A thick line segment corresponding to $\theta_{actual} > 45^{\circ}$

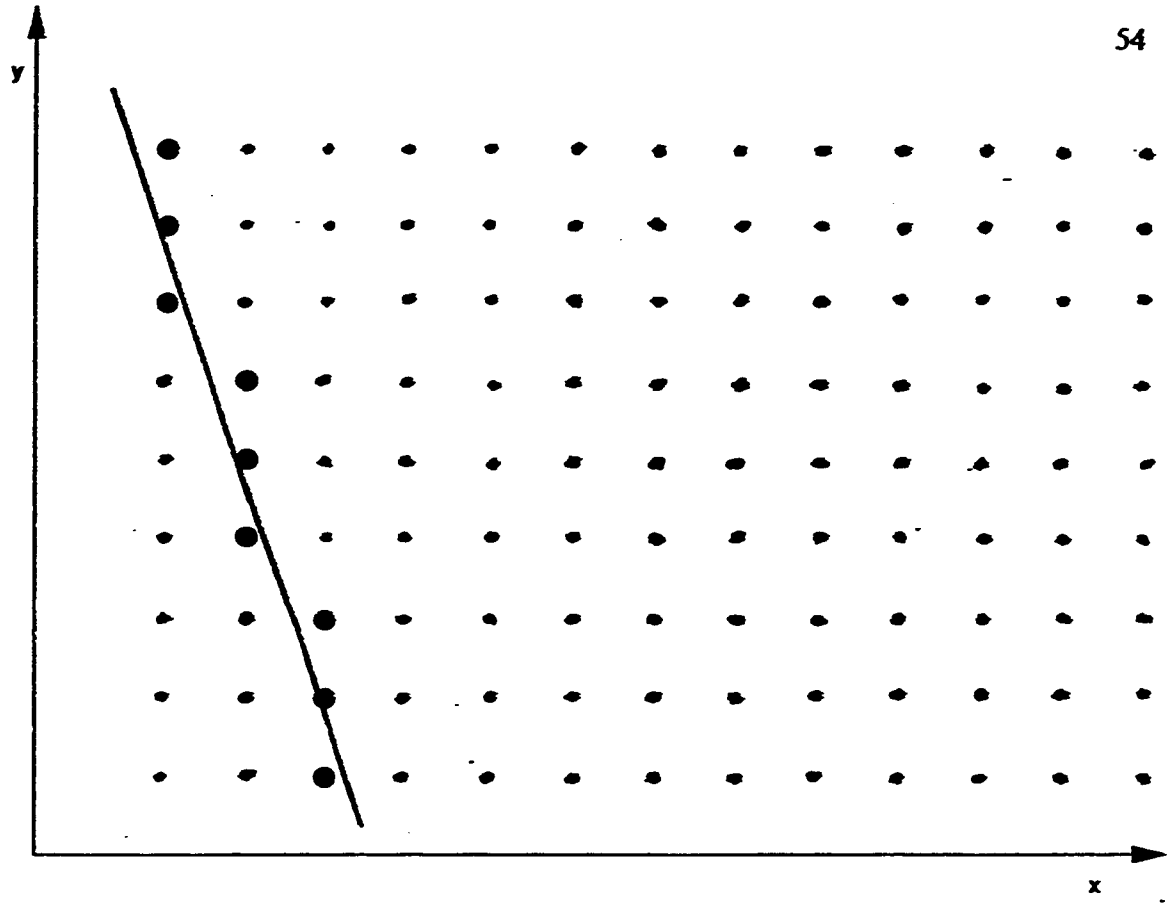


Fig 4.4.2.2 A thick line segment corresponding to $\theta_{actual} < 45^{\circ}$

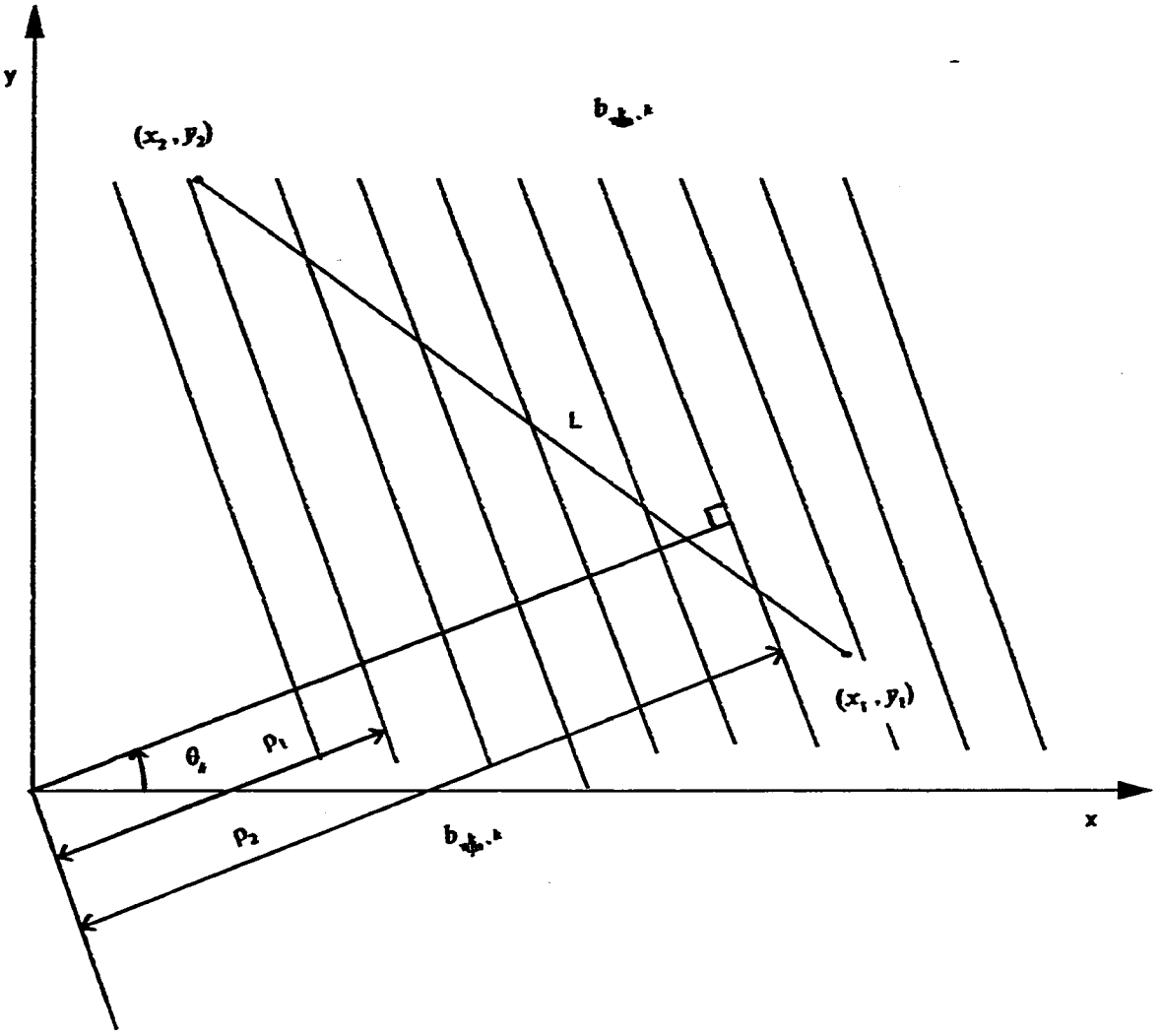


Fig 4.4.2.3 Determination of the endpoints from the spread in the accumulator array using C_k, $k < \theta_{peak}$.

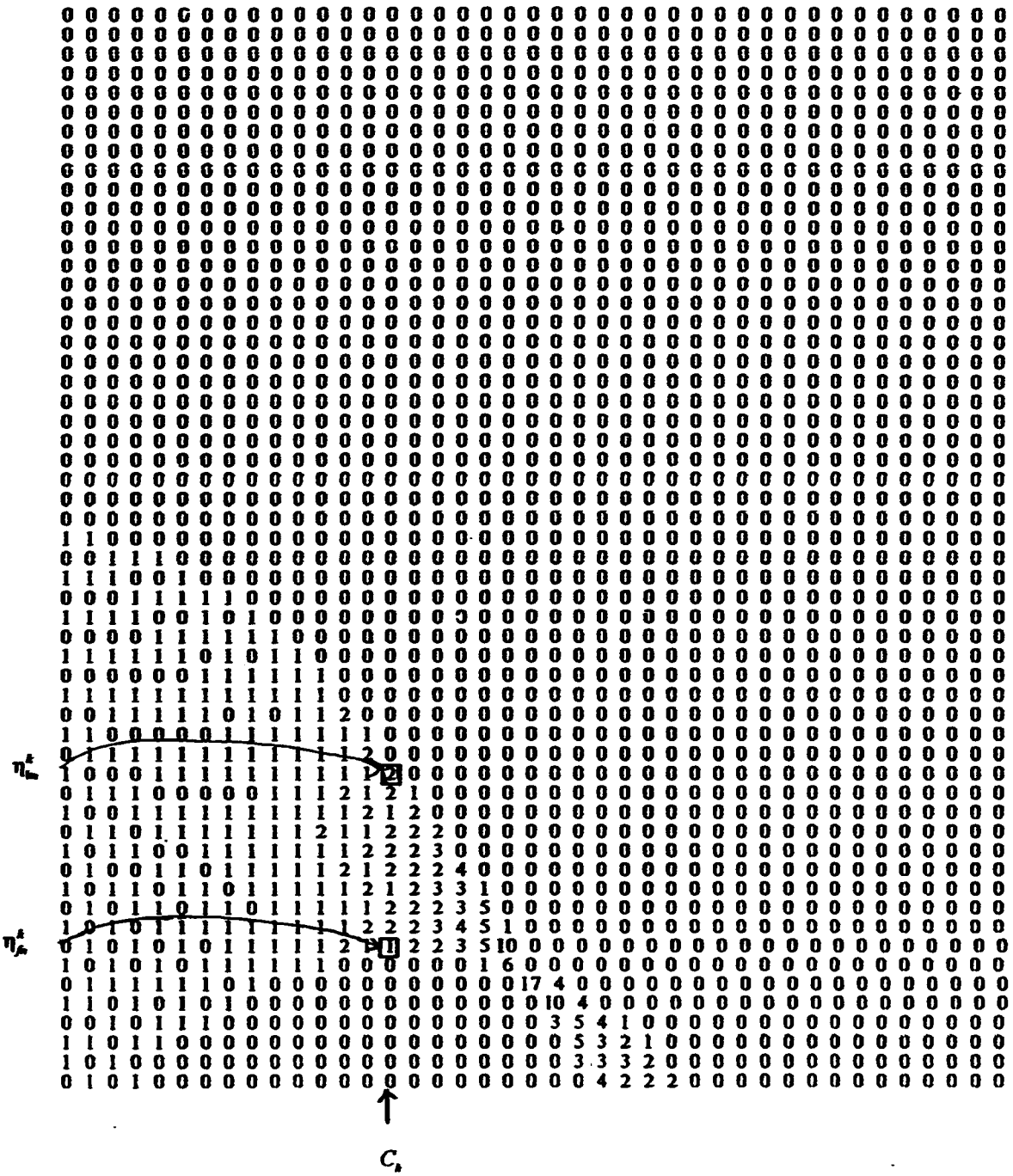


Fig 4.4.2 Part of a typical accumulator array illustrating η^k_{jn} and η^k_{im}

endpoint (x_1, y_1) which lies in $b_{k,k}$ as shown in fig 4.4.2.3. The angle of the normal to the bars in C_k is given by

$$\theta_k = \theta_{predicted} + d_{k,\theta_{peak}}\Delta\theta \qquad 4.19$$

The two bars corresponding to the first and last non-zero entries can be respectively expressed in normal form as

$$\rho_1 = x\cos(\theta_{predicted} + d_{k,\theta_{peak}}\Delta\theta) + y\sin(\theta_{predicted} + d_{k,\theta_{peak}}\Delta\theta) \qquad 4.20$$

$$\rho_2 = x\cos(\theta_{predicted} + d_{k,\theta_{peak}}\Delta\theta) + y\sin(\theta_{predicted} + d_{k,\theta_{peak}}\Delta\theta) \qquad 4.21$$

Using the coarsely predicted values of ρ and θ i.e., $\rho_{predicted}$ and $\theta_{predicted}$, the line segment can be expressed as

$$\rho_{predicted} = x\cos\theta_{predicted} + y\sin\theta_{predicted} \qquad 4.22$$

The endpoints (x_1, y_1) and (x_2, y_2) can be determined by solving the set of linear simultaneous equations eq 4.22 & eq 4.21 and eq 4.22 & eq 4.20 respectively. The expressions for the endpoints are

$$x_1 = \frac{(\rho_{predicted} \times \sin(\theta_{predicted} + d_{k,\theta_{peak}}\Delta\theta)) - (\rho_2 \times \sin\theta_{predicted})}{\sin(d_{k,\theta_{peak}}\Delta\theta)} \qquad 4.23.a$$

$$y_1 = \frac{(\rho_{predicted} - x_1\cos\theta_{predicted})}{\sin\theta_{predicted}} \qquad 4.23.b$$

$$x_2 = \frac{(\rho_{predicted} \times \sin(\theta_{predicted} + d_{k,\theta_{peak}}\Delta\theta)) - (\rho_1 \times \sin\theta_{predicted})}{\sin(d_{k,\theta_{peak}}\Delta\theta)} \qquad 4.23.c$$

$$y_2 = \frac{(\rho_{predicted} - x_2\cos\theta_{predicted})}{\sin\theta_{predicted}} \qquad 4.23.d$$

The above expressions hold when column C_k, $k < \theta_{peak}$ is used. For C_k, $k > \theta_{peak}$ the expressions for (x_1, y_1) and (x_2, y_2) in eq 4.23 are interchanged i.e., (x_1, y_1) is given by eq 4.23.c and eq 4.23.d while (x_2, y_2) is given by eq 4.23.a and eq 4.23.b. This is illustrated in fig 4.4.3. This is done because, now the first and last non-zero entries correspond to (x_1, y_1) and (x_2, y_2) respectively as shown in fig 4.4.2.4. From elementary geometry, the length of the line segment is given by

$$l_{computed} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \qquad 4.24$$

Results for synthetic images using the derived equations are shown in fig 5.2.1.1 to fig 5.2.1.3. These results are elaborated in sec 5.2.1.

The accuracy of the computed endpoints depends on $\Delta\rho$ and $\Delta\theta$ and the accuracy of $\rho_{predicted}$ and $\theta_{predicted}$. The computed endpoints may not lie exactly on the line corresponding to ρ_1 and ρ_2, but may be anywhere in the bar of width $\Delta\rho$. For example, in fig 4.4.3.1, P5 and P4 will computed as endpoints by using eq 4.23, whereas the actual endpoints are P1 and P2.

4.4.3 Length and Extremities through Extrapolation

The method discussed in the previous section may give inaccurate results, since the computed endpoints may not necessarily lie exactly on the bar which is

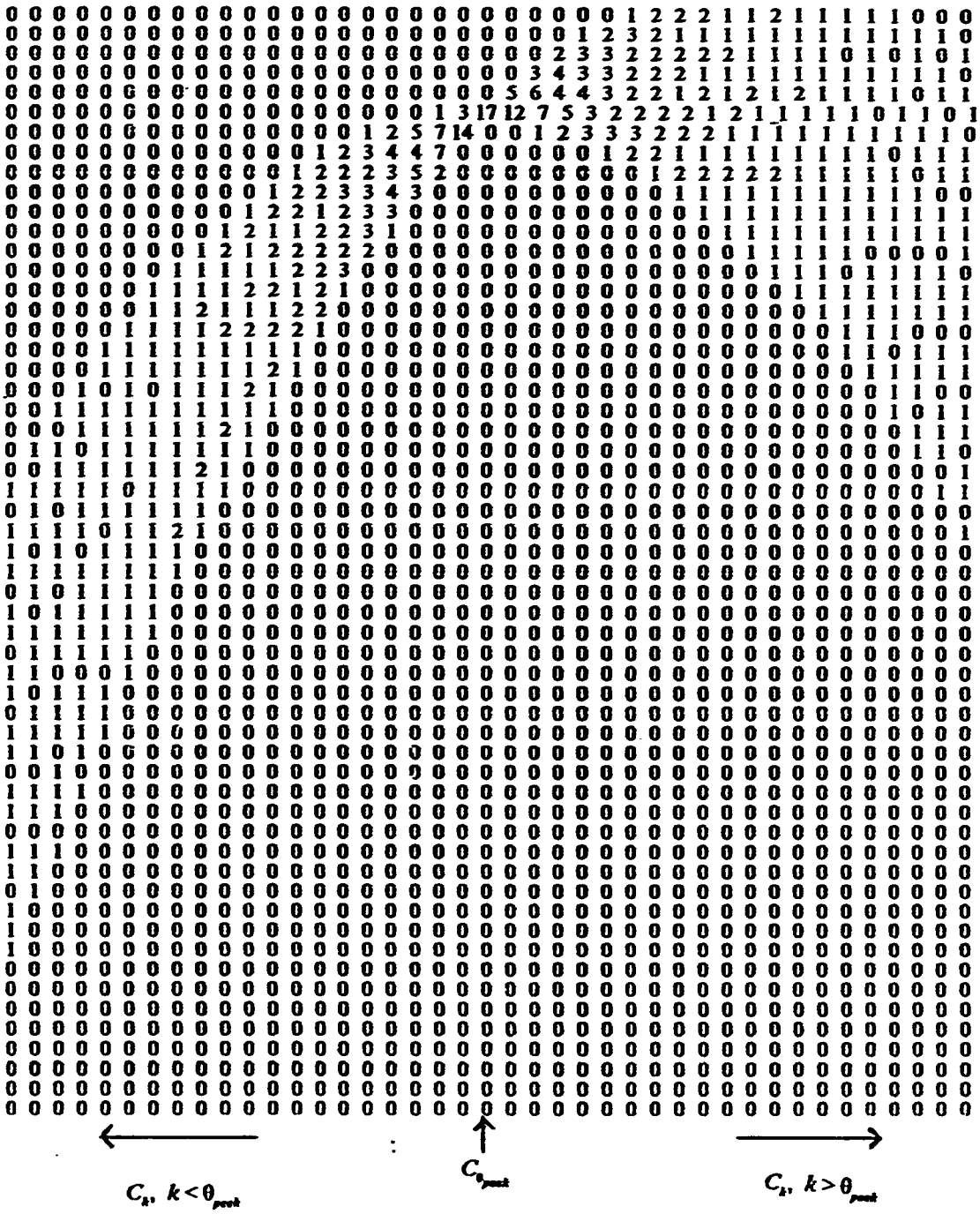


Fig 4.4.3 Part of a typical accumulator array illustrating $C_k, k > \theta_{peak}$ and $C_k, k < \theta_{peak}$

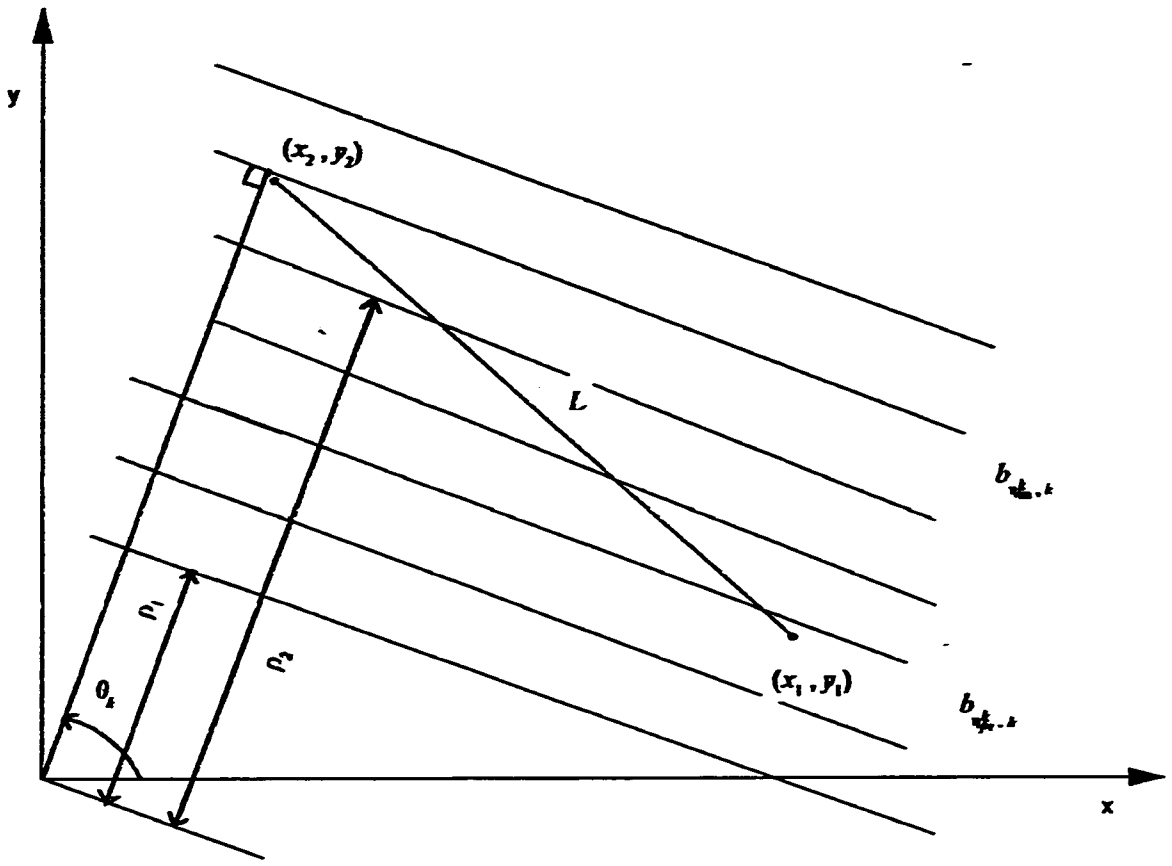


Fig 4.4.2.4 Determination of the endpoints from the spread in the accumulator array using C_k, $k > \theta_{peak}$.

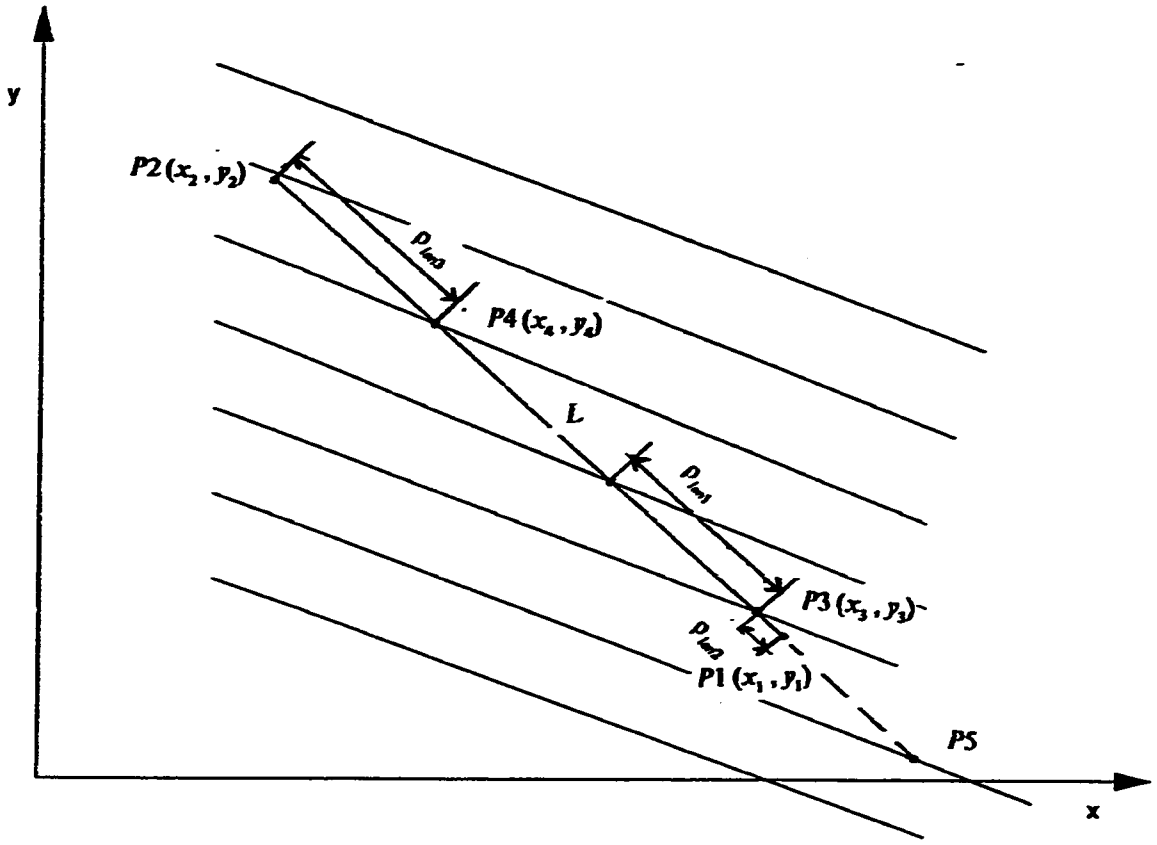


Fig 4.4.3.1 Endpoints detected before and after extrapolation .

used in computation. This drawback can be overcome to a certain extent by using extrapolation. While using extrapolation, initially the points (x_3,y_3) and (x_4,y_4) as shown in fig 4.4.3.1 are computed from expressions similar to eq 4.23 except that now (x_3,y_3) corresponds to the second non-zero entry and (x_4,y_4) to the last but one non-zero entry in the column. (x_3,y_3) is given as the intersection of

$$\rho_3 = x_3 \cos(\theta_{predicted} + d_{k,\theta_{peak}} \Delta\theta) + y_3 \sin(\theta_{predicted} + d_{k,\theta_{peak}} \Delta\theta) \qquad 4.25.a$$

and

$$\rho_{predicted} = x_3 \cos\theta_{predicted} + y_3 \sin\theta_{predicted} \qquad 4.25.b$$

where ρ_3 in eq 4.25.a is given by

$$\rho_3 = \rho_{min} + (\eta_{fn}^k + 1)\,\Delta\rho \qquad 4.26$$

Similarly (x_4, y_4) is the intersection of

$$\rho_4 = x_4 \cos(\theta_{predicted} + d_{k,\theta_{peak}} \Delta\theta) + y_4 \sin(\theta_{predicted} + d_{k,\theta_{peak}} \Delta\theta) \qquad 4.27.a$$

and

$$\rho_{predicted} = x_4 \cos\theta_{predicted} + y_4 \sin\theta_{predicted} \qquad 4.27.b$$

where ρ_4 in eq 4.27.a is given by

$$\rho_4 = \rho_{min} + (\eta_{ln}^k - 1)\,\Delta\rho \qquad 4.28$$

Let ρ_{len1} be the length of the line segment intersected by a bar of width $\Delta\rho$ as shown in fig 4.4.3.1. The length ρ_{len1} in fig 4.4.3.1 is given as

$$\rho_{len1} = \frac{\Delta\rho}{\sin(d_{k,\theta_{pred}}\Delta\theta + \Delta\theta/2)} \qquad 4.29$$

The lengths ρ_{len2} and ρ_{len3} which are defined analogous to ρ_{len1} are determined using the proportion of votes in the corresponding cells of that column in the accumulator array, with the votes in the cell whose corresponding bar is such that the line segment crosses the bar completely.

$$\rho_{len2} = \frac{n_{\theta_{pred}^{k},k}}{n_{\theta_{pred}^{k}+1,k}} \times \rho_{len1} \qquad 4.30.a$$

and

$$\rho_{len3} = \frac{n_{\theta_{pred}^{k},k}}{n_{\theta_{pred}^{k}-1,k}} \times \rho_{len1} \qquad 4.30.b$$

where $n_{i,j}$ is the number of votes or the count in $a_{i,j}$. Using (x_3, y_3), (x_4, y_4), ρ_{len2}, ρ_{len3}, and $\theta_{predicted}$ the endpoints can be determined as shown in fig 4.4.3.1. Use of geometrical and trigonometrical techniques gives the endpoints as

$$x_1 = x_3 + (\sin\theta_{predicted} \times \rho_{len2}) \qquad 4.31.a$$

$$y_1 = y_3 - (\cos\theta_{predicted} \times \rho_{len2}) \qquad 4.31.b$$

$$x_2 = x_4 - (\sin\theta_{predicted} \times \rho_{len3}) \qquad 4.31.c$$

$$y_2 = y_4 + (\cos\theta_{predicted} \times \rho_{len3}) \qquad 4.31.d$$

While employing proportions to determine ρ_{len2} and ρ_{len3} from ρ_{len1}, the votes corresponding to the length ρ_{len1} were taken as $n_{\rho_{len}+1,k}$ or $n_{\rho_{len}-1,k}$; but depending on the value of $\Delta\rho$, there may or may not be zero votes in between non-zero votes in a column. If either $n_{\rho_{len}+1,k}$ or $n_{\rho_{len}-1,k}$ is zero, then division by zero (which is indeterminate) will take place in eq 4.30. This undesirable division by zero was overcome by averaging the counts over a number of cells between $a_{\rho_{len}+1,k}$ and $a_{\rho_{len}-1,k}$ The number of cells having zero votes as entries were excluded or included depending on whether $\Delta\rho$ was large (coarse) or small (fine) respectively. Simulation results using extrapolation are shown in fig 5.2.2.1 to fig 5.5.2.9 which are discussed in detail in sec 5.2.2.

4.4.4 Length and Extremities of a Thick Line Segment

The endpoints of a thick line can also be determined by using the expressions derived for the thin line. But an initial estimate of θ i.e., $\theta_{predicted}$ is necessary. For thick lines, with $\theta_{predicted} > 45^{\circ}$, the endpoints are computed from the spread in C_k, $k < \theta_{peak}$. Computation of endpoints using C_k, $k < \theta_{peak}$, makes sure that the points (x_1, y_1) and (x_2, y_2) are detected and not (x_3, y_3) and (x_4, y_4) as shown in fig 4.4.4.1. If columns C_k, $k > \theta_{peak}$ are used, ambiguity could result in computed endpoints. This is because there is a possibility of (x_1, y_1) and

(x_3, y_3) contributing to the same cell and (x_2, y_2) and (x_4, y_4) contributing to another cell.

For thick lines with $\theta_{predicted} < 45^{\circ}$, the endpoints are computed from the spread in C_k, $k > \theta_{peak}$. This ensures that (x_1, y_1) and (x_2, y_2) are detected and not (x_3, y_3) and (x_4, y_4) as shown in fig 4.4.4.2. If column C_k, $k < \theta_{peak}$, is used then ambiguity occurs in the computed endpoints as discussed before.

Extrapolation techniques can also be applied to the thick line segments for enhancing the accuracy of the computed endpoints. The length of the line segment is computed from the extremities using eq 4.24.

4.4.5 Drawbacks of the approaches discussed

The procedure to compute the endpoints, discussed in the foregoing sections has several drawbacks, especially for the thick lines which are inevitably present in real world images. In a digitized line there can be several consecutive feature points having one of the coordinates equal. It is possible that due to the inclination and quantization of the line, a single feature point may be present at the end of the line segment as shown in fig 4.4.5.1. The probability that this single endpoint is detected using the approach discussed is very low. This is because if column C_k such that k is close to θ_{peak} is used, then the first non-zero entry in C_k will not be due to the contribution of this isolated endpoint. It is

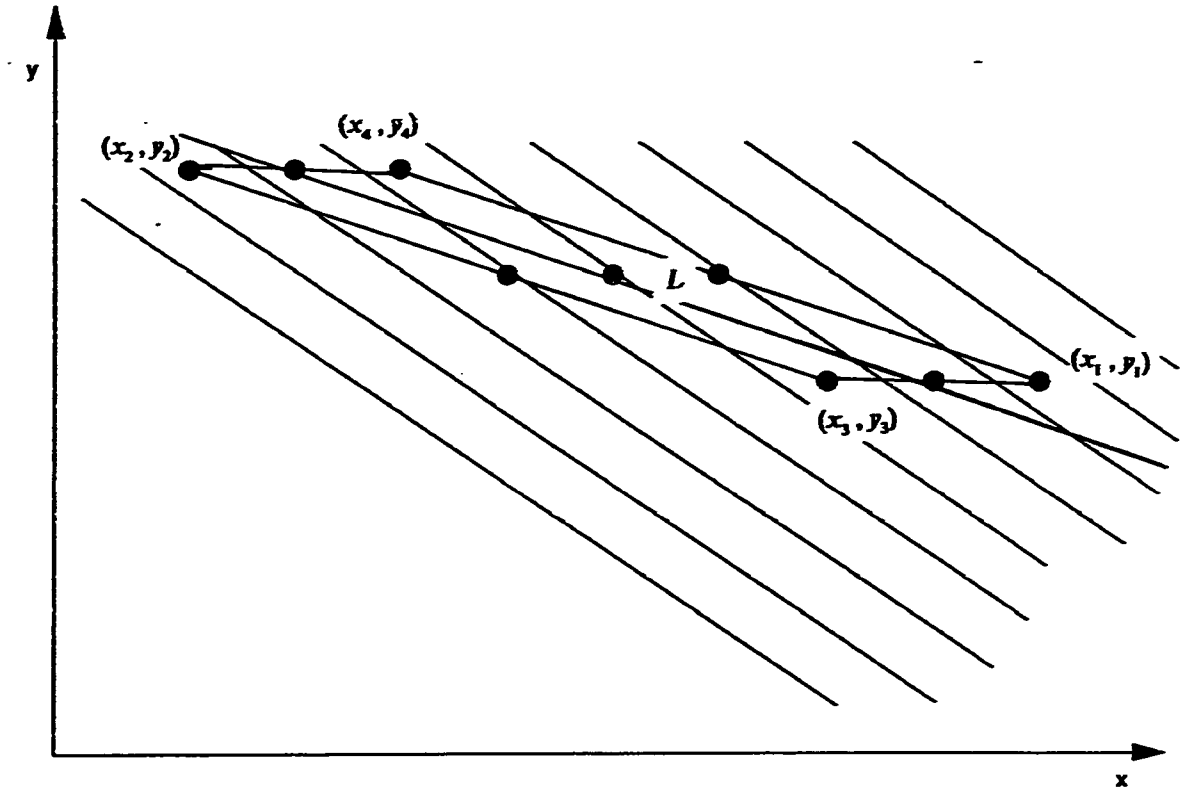


Fig 4.4.4.1 Extremeties of a thick line segment corresponding to $\theta_{actual} \geq 45^{\circ}$.

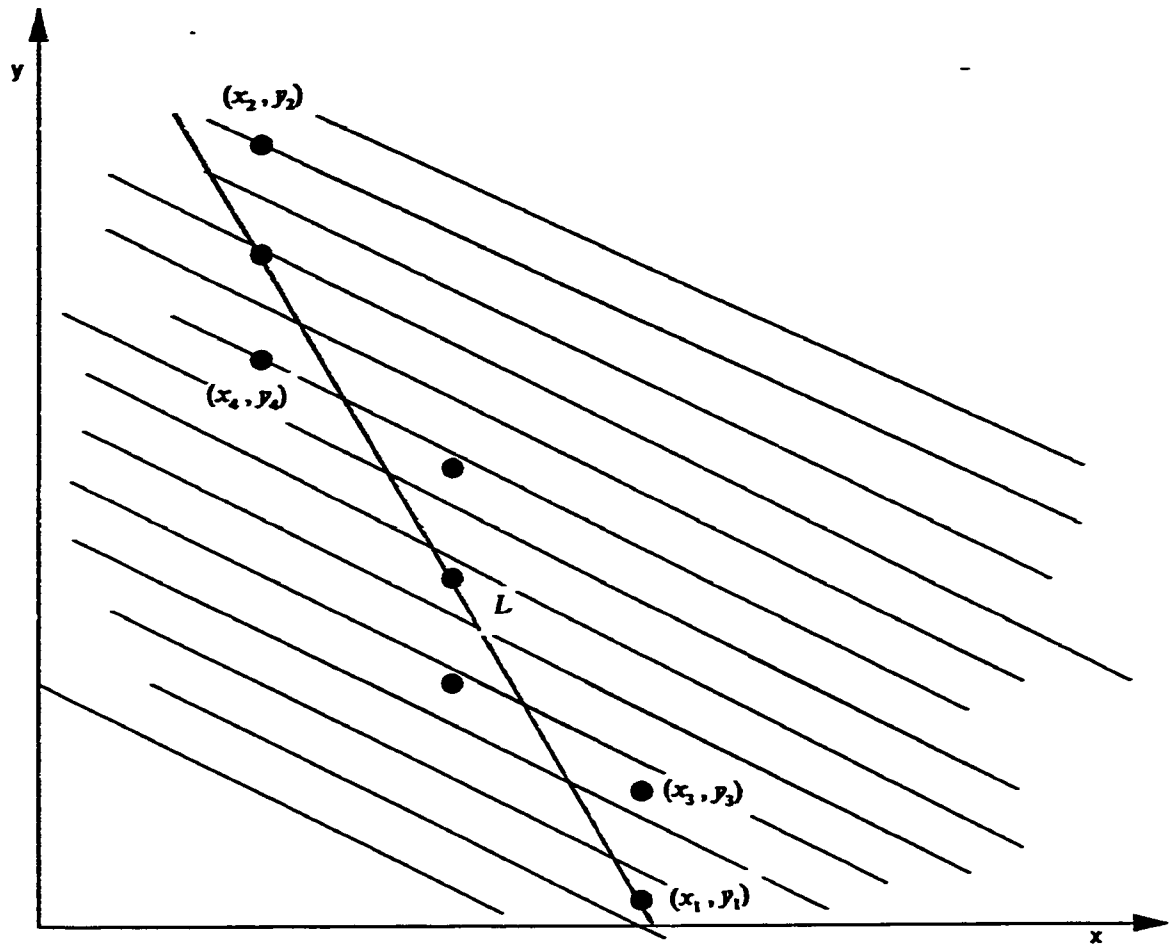


Fig 4.4.4.2 Extremeties of a thick line segment corresponding to $\theta_{actual} < 45^{\circ}$.

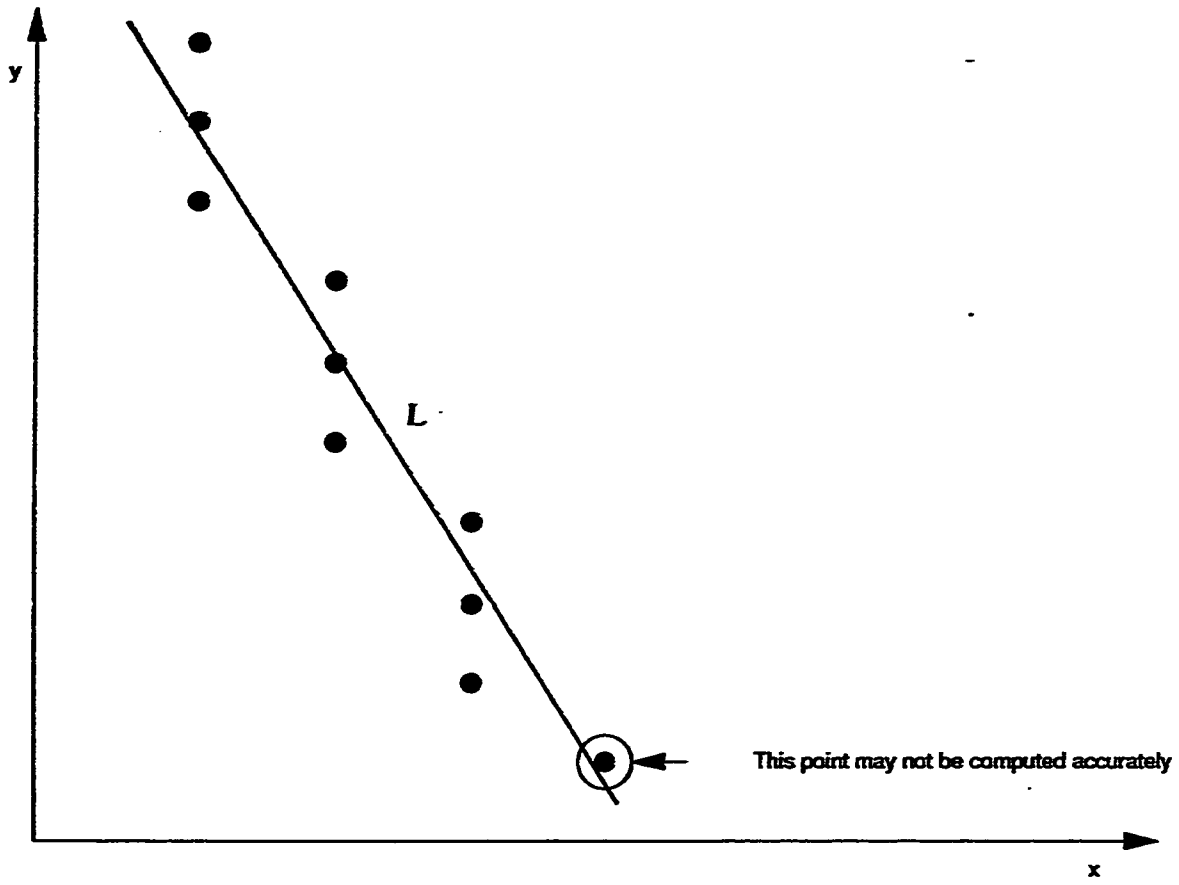


Fig 4.4.5.1 An isolated feature point which may not be detected accurately.

also possible that the predicted line segment and the bars used in the computation of endpoints may not intersect within the image boundaries. The intersection point will then lie outside the image plane while the endpoints lie within the image plane. Another important drawback is that, $\theta_{predicted}$ is used in determining the endpoints. The accuracy of the computed endpoints depends directly on the accuracy of $\theta_{predicted}$. An algorithm which overcomes the above discussed drawbacks is presented in the next section.

4.5 Complete Line Segment Description without using $\theta_{predicted}$

In the foregoing sections, the extremities and length of L were computed by using $\theta_{predicted}$. But failure to detect the peak properly could lead to inaccurate value of $\theta_{predicted}$. In this section, an approach which utilizes $\theta_{predicted}$ as a condition rather than actually using it in the computation of the extremities of the line segment is presented. The computation of the extremities can be made independent of $\theta_{predicted}$ by eliminating eq 4.22 from the set of linear simultaneous equations that are solved for determining the extremities. This can be done in the following way (for both thin as well as thick line segments)as shown in fig 4.5.1 and fig 4.5.2.

Consider two columns C_q and C_r whose cells in the accumulator array correspond to bars in the image plane with their normals inclined at angles θ_q

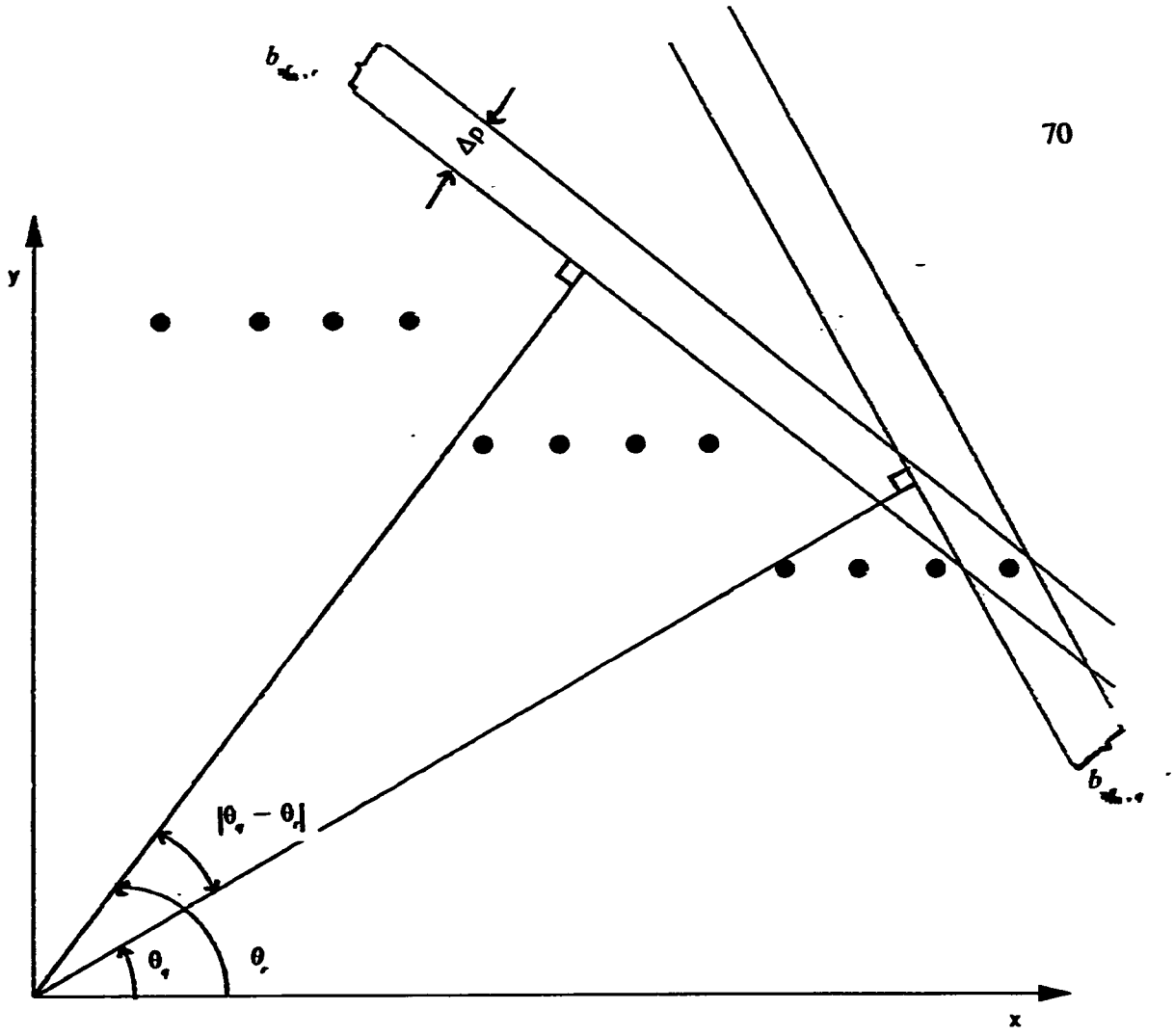


Fig 4.5.1 Computation of the extremeties, independent of $\theta_{predicted}$ for $\theta_{predicted} \geq 45^{\circ}$.

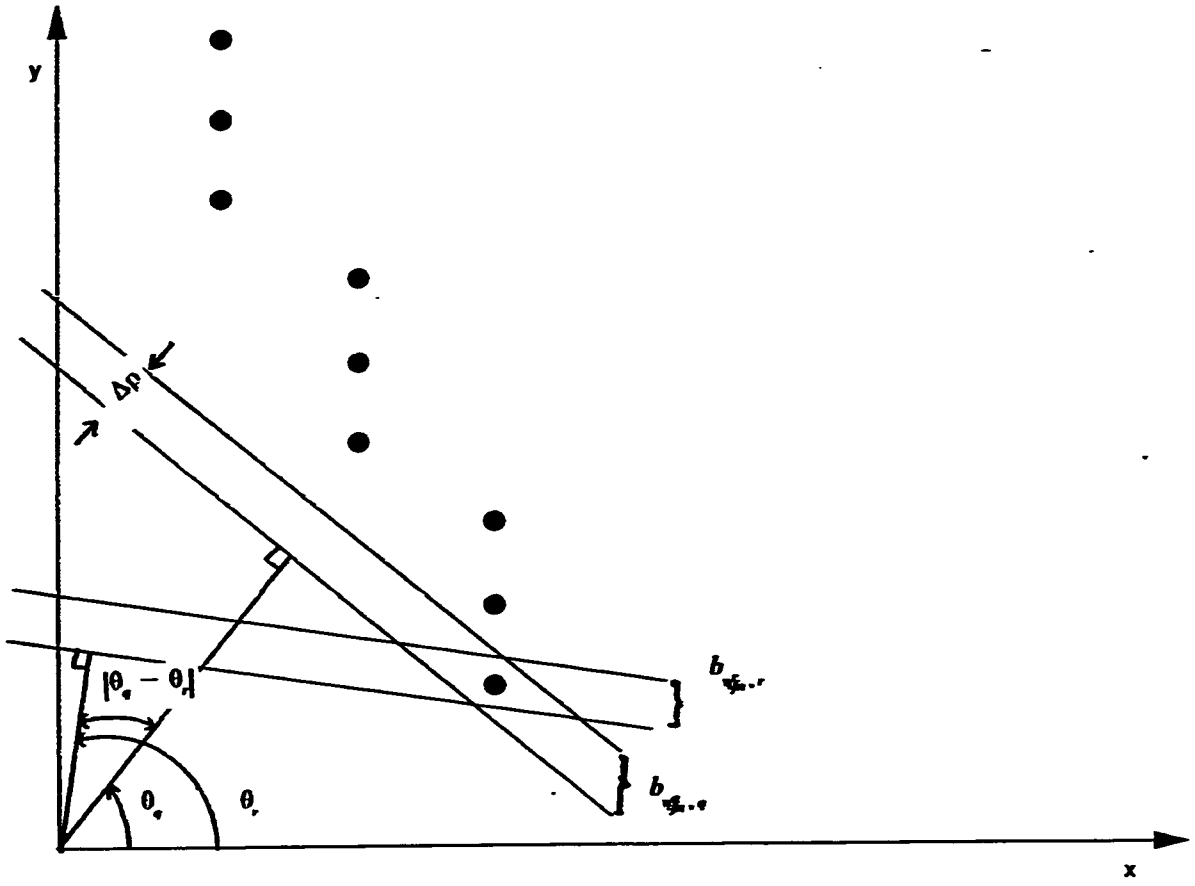


Fig 4.5.2 Computation of the extremeties, independent of $\theta_{predicted}$ for $\theta_{predicted} < 45^{\circ}$.

and θ_r respectively such that $\theta_q \neq \theta_r \neq \theta_{actual}$ ($\theta_q = \theta_{min} + q\,\Delta\theta$, $\theta_r = \theta_{min} + r\,\Delta\theta$). Determine η^q_{fs}, η^r_{fs}, η^q_{ls}, η^r_{ls}, from A and compute the lengths of the normals corresponding to η^q_{fs}, η^q_{ls}, η^r_{fs}, η^r_{ls}, respectively as follows

$$\rho^q_1 = \rho_{min} + \eta^q_{fs}\,\Delta\rho + \frac{\Delta\rho}{2} \qquad 4.32.a$$

$$\rho^q_2 = \rho_{min} + \eta^q_{ls}\,\Delta\rho + \frac{\Delta\rho}{2} \qquad 4.32.b$$

$$\rho^r_1 = \rho_{min} + \eta^r_{fs}\,\Delta\rho + \frac{\Delta\rho}{2} \qquad 4.32.c$$

$$\rho^r_2 = \rho_{min} + \eta^r_{ls}\,\Delta\rho + \frac{\Delta\rho}{2} \qquad 4.32.d$$

These bars can be expressed in the normal form as

$$\rho^q_1 = x_1 \cos\theta_q + y_1 \sin\theta_q \qquad 4.33.a$$

$$\rho^r_1 = x_1 \cos\theta_r + y_1 \sin\theta_r \qquad 4.33.b$$

and

$$\rho^q_2 = x_2 \cos\theta_q + y_2 \sin\theta_q \qquad 4.34.a$$

$$\rho^r_2 = x_2 \cos\theta_r + y_2 \sin\theta_r \qquad 4.34.b$$

By substituting the values of ρ^q_1, ρ^r_1, ρ^q_2, ρ^r_2 from eq 4.32 in eq 4.33 and eq 4.34, (x_1, y_1) and (x_2, y_2) calculated by solving equation sets 4.33 and 4.34 respectively. The expressions for the computed co-ordinates are

$$x_1 = \frac{\rho_1^q \sin\theta_r - \rho_1^r \sin\theta_q}{\sin(\theta_r - \theta_q)} \qquad 4.35.a$$

$$y_1 = \frac{\rho_1^r \cos\theta_q - \rho_1^q \cos\theta_r}{\sin(\theta_r - \theta_q)} \qquad 4.35.b$$

and

$$x_2 = \frac{\rho_2^q \sin\theta_r - \rho_2^r \sin\theta_q}{\sin(\theta_r - \theta_q)} \qquad 4.36.a$$

$$y_2 = \frac{\rho_2^r \cos\theta_q - \rho_2^q \cos\theta_r}{\sin(\theta_r - \theta_q)} \qquad 4.36.b$$

The length is determined using eq 4.24. As stated earlier for lines with $\theta_{predicted} < 45^\circ$, columns C_q, C_r, $q > \theta_{peak}$, $r > \theta_{peak}$ are used and for lines having $\theta_{predicted} \geq 45^\circ$, columns C_q, C_r, $q < \theta_{peak}$, $r < \theta_{peak}$ are used. This is necessary to remove the ambiguity that could result if the columns are chosen in the opposite directions. This ambiguity has been explained in sec. 4.4.4. In order to increase the accuracy of the computed endpoints, a $\frac{\Delta\rho}{2}$ term has been used in equation set 4.32. The maximum error that could result in the computed x and y co-ordinates of the endpoints will be denoted by ε_x and ε_y. These errors are given by

$$\varepsilon_x = \frac{\frac{\Delta\rho}{2} \sin(\frac{\theta_q + \theta_r}{2})}{\sin(\frac{\theta_r - \theta_q}{2})} \qquad 4.37$$

and

$$\varepsilon_y = \frac{\frac{\Delta\rho}{2}\cos\left(\frac{\theta_q + \theta_r}{2}\right)}{\sin\left(\frac{\theta_r - \theta_q}{2}\right)} \qquad 4.38$$

From the above equations it is seen that the error in computed co-ordinates is a function of $\Delta\rho$ and $d_{q,r}\Delta\theta$. As $d_{q,r}\Delta\theta$ increases, the error decreases (since the sine of an angle increases with a increase in the angle). Hence the accuracy of the computed endpoints can be increased by increasing the angle between the normals to the bars. This can be done by either

- increasing $\Delta\theta$ which could lead to a decrease in the computational complexity (which is very much desired) at the expense of the accuracy of $\rho_{predicted}$ and $\theta_{predicted}$. But this loss in accuracy of $\rho_{predicted}$ and $\theta_{predicted}$ can be overcome by computing ρ and θ from the extremities as $\rho_{computed}$ and $\theta_{computed}$ using eq 4.41 and eq 4.40 respectively.

or by

- solving for two columns C_q and C_r such that $|q - r|$ is large.

Results using synthetic images containing single line segments are shown in fig 5.3.1 to fig 5.3.16 and are explained in sec 5.3.

The principal advantage of this approach is that $\theta_{predicted}$ is not used in the actual computation of the line extremities. It is only used as a condition which is tested, to select columns to be used in the computation of the extremities. The columns C_k, $k < \theta_{peak}$ are selected if $\theta_{predicted} \geq 45^{\circ}$ and those with $k > \theta_{peak}$ are selected if $\theta_{predicted} < 45^{\circ}$.

4.5.1 Optimal Accumulator Array Size

In real-world images, most of the images encountered contain thick lines. In order to increase the accuracy of $\rho_{predicted}$ and $\theta_{predicted}$ the resolutions $\Delta\rho$ and $\Delta\theta$ are made small which results in undesirable increase in memory space requirements and computational complexity of the HT. For thick lines, a decrease in $\Delta\rho$ could result in two types of patterns in the accumulator array, which are undesirable from the view-point of the peak searching algorithm. These are

1. A pattern of almost equal entries along any column C_k, such that consecutive cells within that column differ by one vote, as shown in fig 4.4.4. This could mislead the peak detection algorithm. Failure to accurately detect the peak will result in high errors in the values of $\rho_{predicted}$ and $\theta_{predicted}$.

2. If the line is too thick and the feature points are such that, a bar of

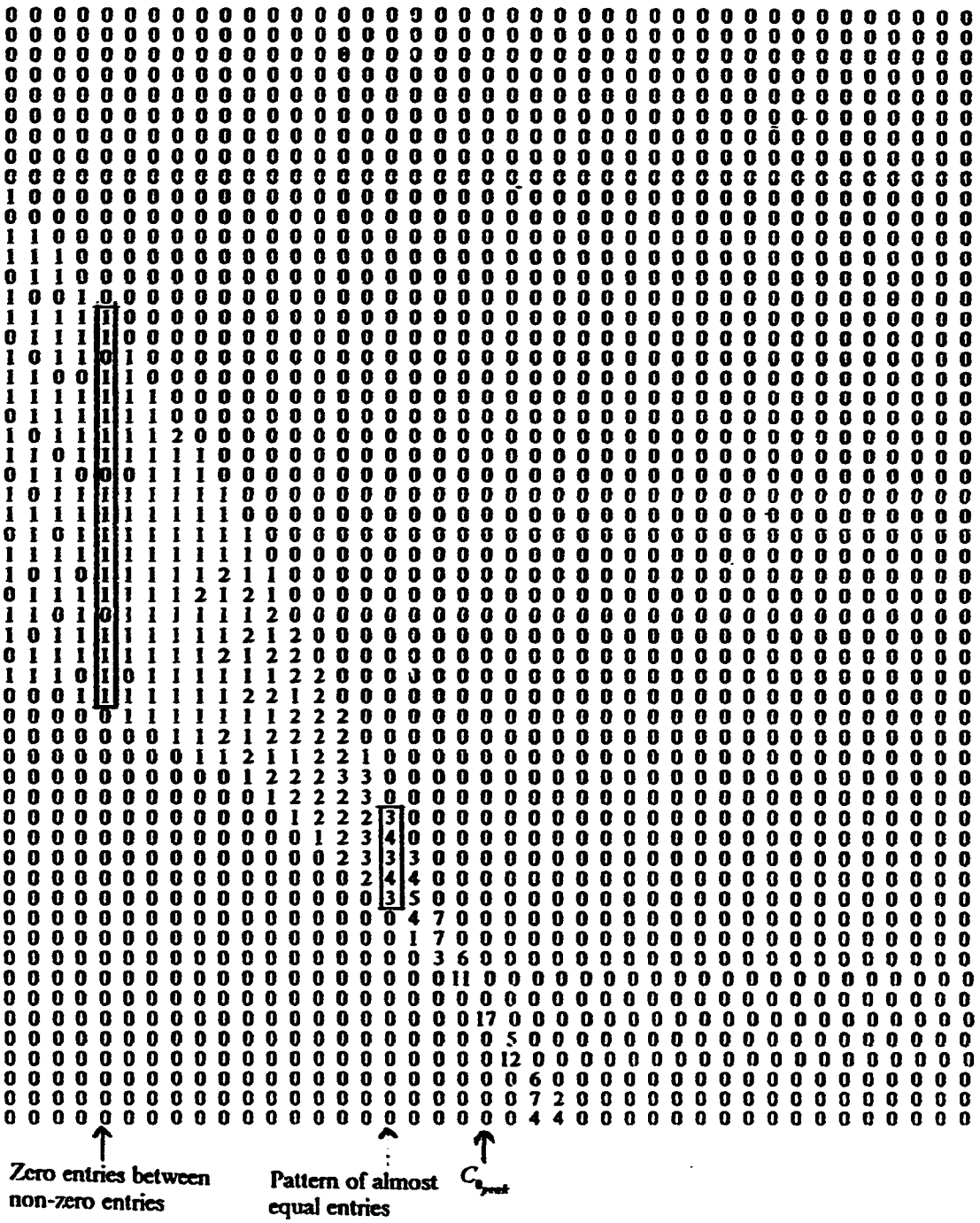


Fig 4.4.4 Part of a typical accumulator array illustrating the pattern of almost equal entries and zero entries between non-zero entries in C_k

width Δρ passes without enclosing any feature points as shown in fig 4.5.1.1, then a regular pattern of cells containing zero votes between non-zero votes will result, which is illustrated in fig 4.4.4. This is particularly undesirable in $C_{\theta_{pred}}$.

The above two undesirable features in the accumulator array of the HT can be avoided by using a value of Δρ such that, a bar of width Δρ includes the maximum number of feature points on the line as shown in fig 4.5.1.2. This means that Δρ has to be increased which leads to a loss in accuracy of $\rho_{predicted}$. But accurate ρ can also be obtained, once the endpoints of the line segment are detected accurately. Hence, Δρ can be increased to avoid the previously described undesirable patterns. How large Δρ should be, depends on the number of feature points in the image line having one of the co-ordinates equal. An estimate of this can be obtained by using $\theta_{predicted}$, from which, the approximate slope $m_{predicted}$ of the line can be obtained using eq 3.10. The number of feature points having one of their co-ordinates equal f_{some} is given by eq 3.11. Van veen & Groen [a1] have stated that Δρ should be atleast d_m, where d_m is given by eq 3.8. But this value of Δρ would also lead to the phenomenon of regular structure of zero votes in the array, if the line segment in the image has a high value of f_{some}. But if Δρ is chosen such that, a bar of width Δρ contains the maximum number of feature points, then the above drawback will be overcome. This can be achieved by using a value of Δρ given by

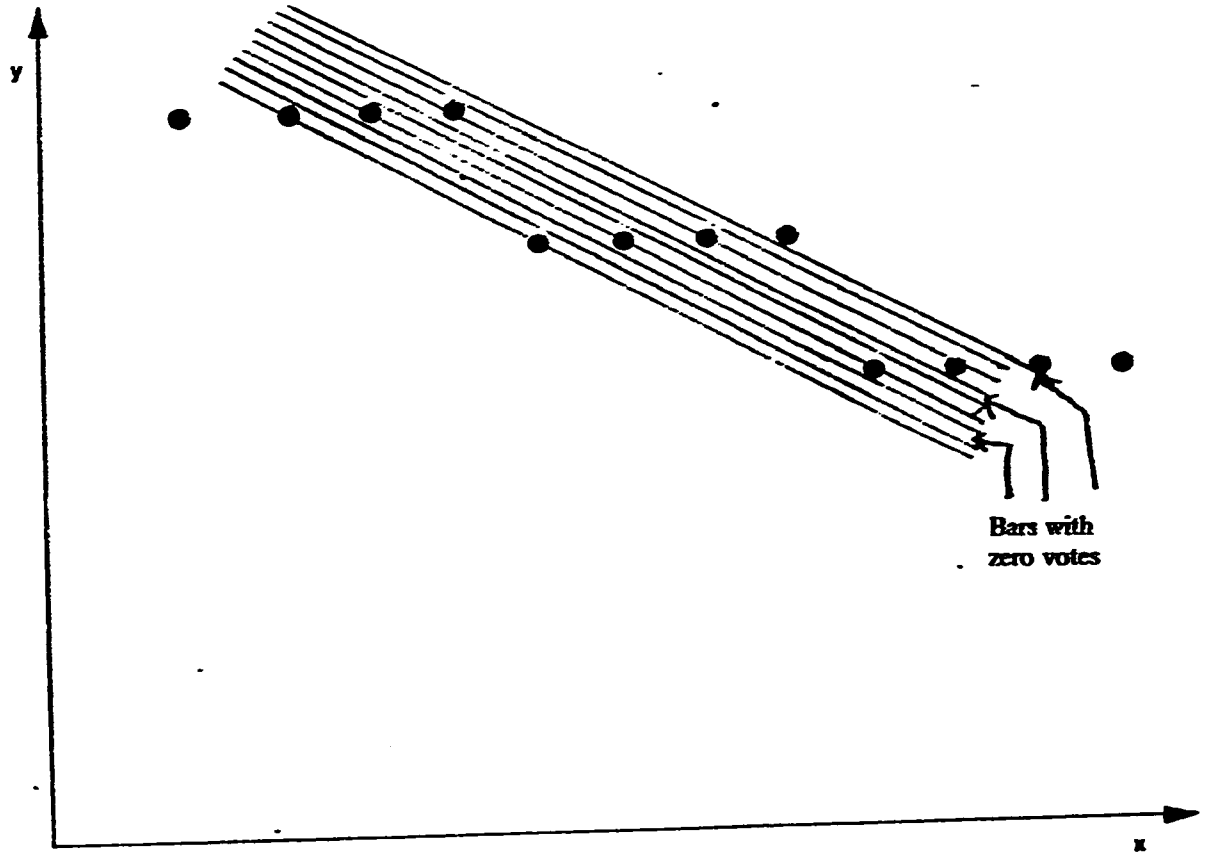


Fig 4.5.1.1 The phenomenon of zero votes in between non-zero votes in the accumulator array.

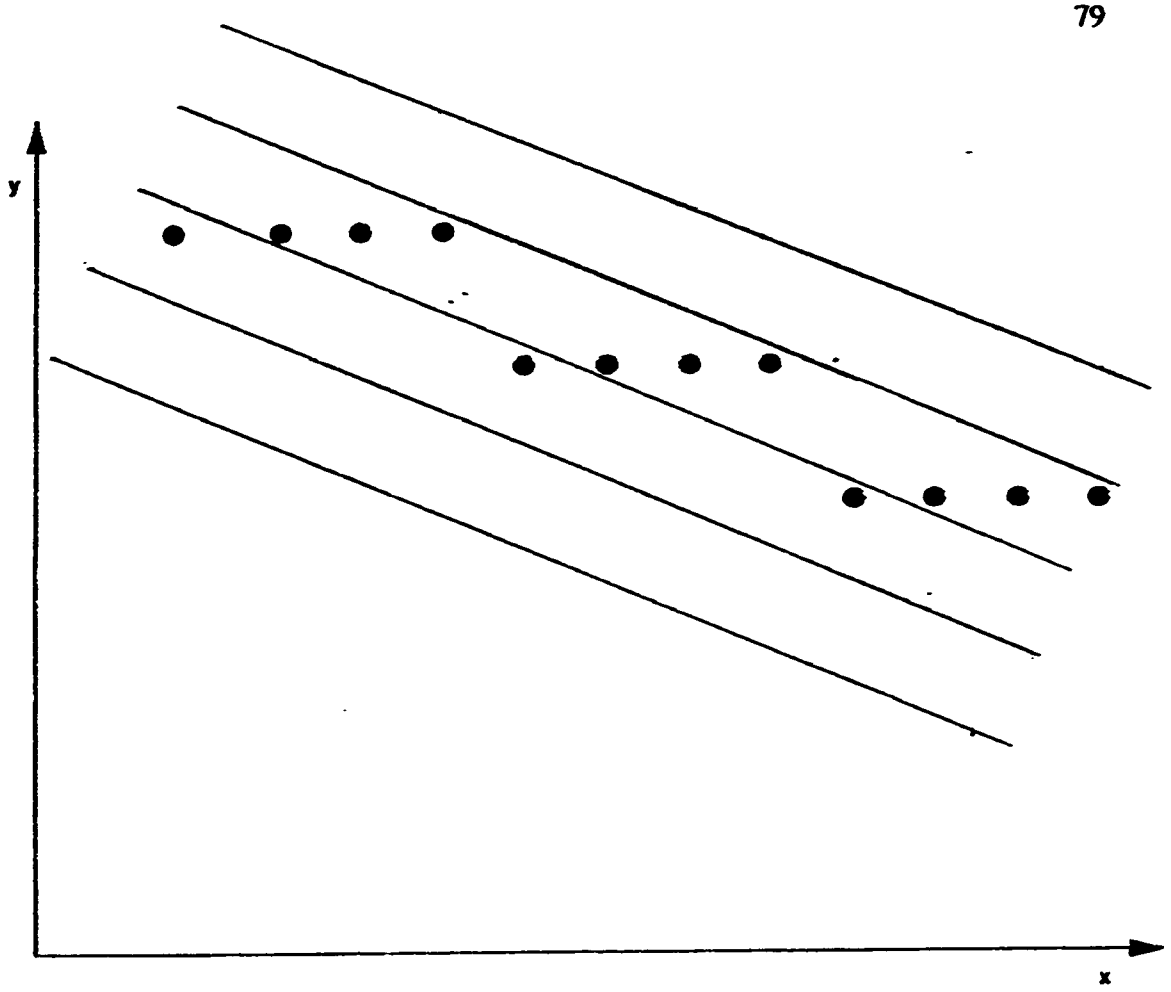


Fig 4.5.1.2 Bars of width Δp, ensuring maximum number of feature points within 2 or 3 such bars.

$$\Delta\rho = \left\lceil \frac{f_{same}}{2} \right\rceil \times d_m \qquad 4.39$$

An approximate value of f_{same} can be obtained from $\theta_{predicted}$ and then the $\Delta\rho$ given by eq 4.39 can be used in subsequent accumulation. The information in the accumulator array with this $\Delta\rho$ can be used to determine the extremities of the line segment in the image plane. The normal parameters of the line segment can be determined from the computed endpoints. If (x_1, y_1) and (x_2, y_2) are the endpoints of L, the inclination θ of the normal to L is given by

$$\theta_{computed} = \tan^{-1}\left(\frac{y_2 - y_1}{x_2 - x_1}\right) - 90^{\circ} \qquad 4.40$$

and the length ρ of the normal is computed as

$$\rho_{computed} = \frac{x_2 \times y_1 - x_1 \times y_2}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}} \qquad 4.41$$

For lines having $\theta_{predicted} \geq 45^{\circ}$, with $\Delta\rho$ chosen as given by eq 4.39 a pattern of zero votes in between non-zero votes will result in A, in columns C_k where $k > \theta_{peak}$. However, this will not cause any accuracy problems, since for such lines, columns C_k having $k < \theta_{peak}$ are used in the computation of the co-ordinates. A similar pattern of zero votes between non-zero votes in columns C_k having $k < \theta_{peak}$ will result for lines with $\theta_{predicted} < 45^{\circ}$. But then, for such lines, columns C_k, $k > \theta_{peak}$, are used for determining the co-ordinates of the endpoints.

The principal advantage of the suggested value of $\Delta\rho$ (eq 4.39) is that, $C_{\rho_{peak}}$ will have a maximum spread of 2 or 3 cells. This will result in unique peak and accurate computed values of the extremities of the line segment, thereby giving highly accurate $\rho_{computed}$ and $\theta_{computed}$.

4.5.2 Detection of Multiple Line Segments

Though images with multiple line segments are not considered in this work, a procedure is suggested, which has to be exhaustively tested. For images containing multiple line segments, the Hough accumulator array is constructed using the conventional approach for the HT. Detection of peaks can be carried out by applying a threshold τ_p to the cells in A. All those cells $a_{i,j}$ having $n_{i,j} \geq \tau_p$ are taken as cells corresponding to the indices of the parameters of different line segments in the image plane. The extremities of different line segments are computed individually by using the procedure for single line segment. Reasonably accurate values of ρ and θ are obtained from the computed co-ordinates. The value of τ_p has to be chosen depending on the quality of the image and the type of the noise present in the image.

4.6 Complete Line Segment Description in presence of noise

In the preceding sections, the proposed method for obtaining the complete line segment description from the spread in the accumulator array was explained. The proposed approach was initially developed for detecting ideal images i.e., noise free images. But in real world images noise is inevitably present in the images. As explained in earlier chapters, the HT is applied after the image is passed through several stages of electronic instruments such as filters, edge detectors etc. These electronic systems also add noise to the image which is finally processed by the HT. But HT is inherently robust against noise problems. This is true when HT is used only for predicting the values of the normal parameters of the line segment. But when the proposed method is used for obtaining the complete line segment description in presence of noise, then considerable amount of errors result. This is because in computing the extremities of the line segment from the spread in the accumulator array, large values of $\Delta\rho$ and $\Delta\theta$ were used and under these circumstances noise points could get detected as the extremities of the line segment.

In this section, a modification is suggested to the method described in sec 4.5 for efficiently computing the extremities of the line segment in presence of random noise. The underlying principle of the method described in sec 4.5 still holds i.e., once the extremities of the line segment are computed accurately, the ρ and θ of the line segment can be computed using eq 4.41 and eq 4.40 respectively.

4.6.1 Effect of noise on the spread in the accumulator array

Noise affects the spread in the accumulator array. Isolated groups of non-zero entries in the accumulator array result due to the presence of noise. In the new approach described in sec 4.5, for computing the extremities of the line segment, columns C_q and C_r in A are chosen depending on the value of $\theta_{predicted}$. In these columns, the values of η^q_{fir} ,η^q_{lin}, η^r_{fir} ,η^r_{lin} corresponding to the contributions from the actual line segment have to be determined accurately. Determination of η^q_{fir} ,η^q_{lin}, η^r_{fir} ,η^r_{lin} as explained in sec 4.5 could lead to considerable errors since these could be due to the contributions from the noise pixels. The methodology adopted to overcome this misleading contributions is to scan columns C_q and C_r for η^q_{fir} ,η^q_{lin}, η^r_{fir} ,η^r_{lin} not from 0 to $\rho_{asize} - 1$, but from cells containing votes, which are contributions of the feature points. A method to achieve this objective is explained in the next section.

4.6.2 Methodology to overcome the effect of noise

Find the value of ρ^k_{startl} and ρ^k_{starth} in any C_k as shown in fig 4.6.1. This is done by finding the value of the co-ordinates of the point of intersection of the predicted line segment with the image boundaries (y-axis if $\theta_{predicted} \geq 45^{\circ}$, x-axis if $\theta_{predicted} < 45^{\circ}$). From these intersections, the values of ρ^k_{startl} and ρ^k_{starth} can be calculated by using the following equations:

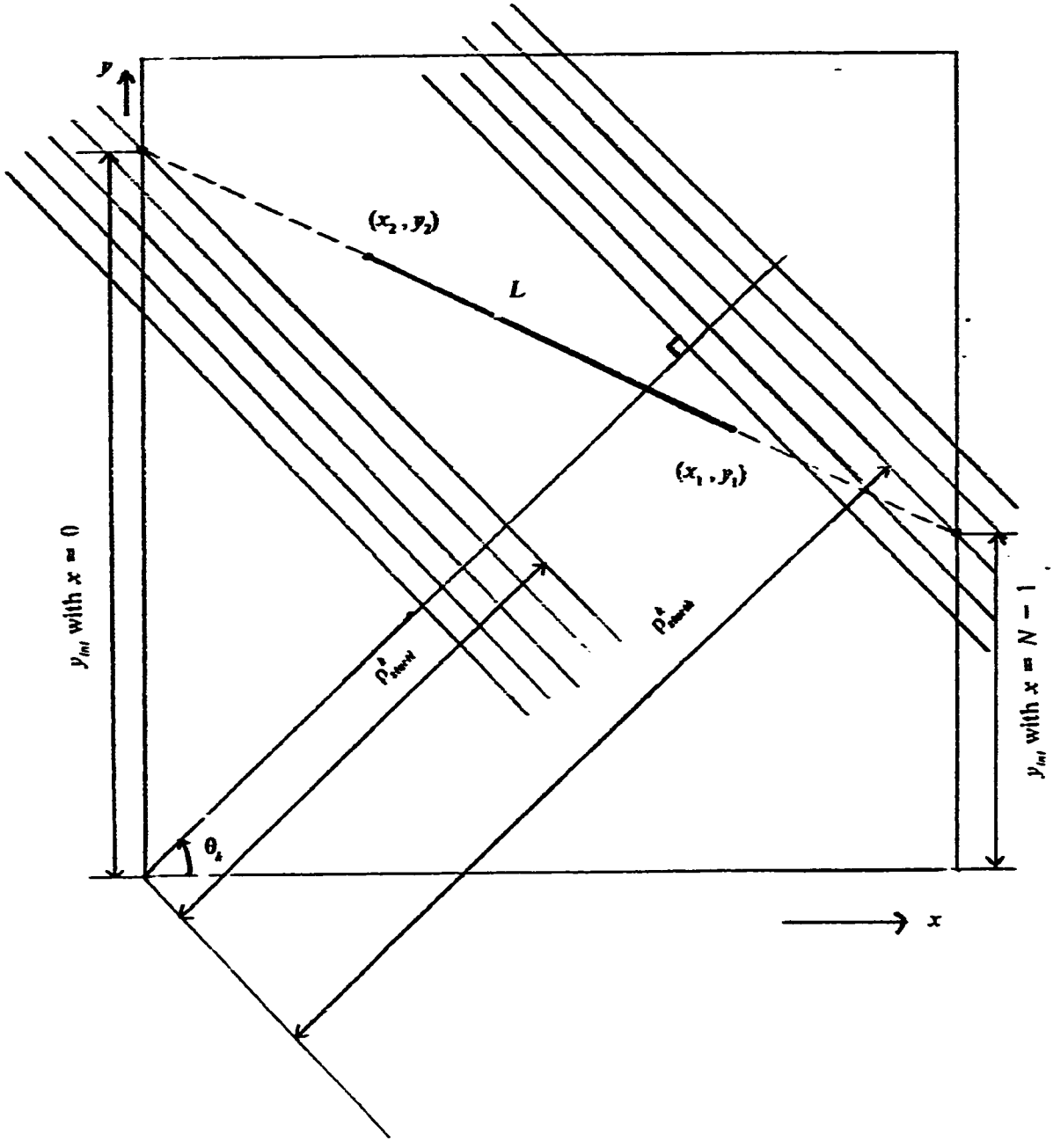


Fig 4.6.1 Determination of ρ^k_{short} and ρ^k_{short} to overcome the effect of noise

$$\rho^{k}_{startl} = x_{int}\cos\theta_k + y_{int}\sin\theta_k \qquad 4.42$$

$$\rho^{k}_{starth} = x_{int}\cos\theta_k + y_{int}\sin\theta_k \qquad 4.43$$

In eq 4.42, $x_{int} = 0$ and y_{int} is the value of y obtained from eq 4.22 by substituting $x = 0$. Similarly in eq 4.43 $x_{int} = N-1$, where N is the image size (in our simulations we used $N = 128$) and y_{int} is the value of y obtained from eq 4.22 by substituting $x = N-1$. These are valid when $\theta_{predicted} \geq 45^{\circ}$. For $\theta_{predicted} < 45^{\circ}$, the x's and y's in the foregoing statements have to be interchanged (for obtaining the intersection with the x-axis).

After ρ^{k}_{startl} and ρ^{k}_{starth} are computed, the value of the corresponding indices in the ρ − direction between which the scanning for non-zero entries is done are evaluated. These indices are obtained by dividing ρ^{k}_{startl} and ρ^{k}_{starth} by $\Delta\rho$ and quantizing the result to the nearest integer to obtain the indices in the ρ − direction, of the cells in C_k. Due to this quantization, some errors could creep in the values of the indices in the ρ − direction between which the scanning is done. This can be overcome by correcting the indices as

$$\eta^{k}_{sl} = int(\frac{\rho^{k}_{startl}}{\Delta\rho}) - \eta^{k}_{correc} \qquad 4.44.a$$

and

$$\eta^{k}_{sh} = int(\frac{\rho^{k}_{starth}}{\Delta\rho}) + \eta^{k}_{correc} \qquad 4.44.b$$

where η^k_{correc} is the number of cells subtracted or added for accounting for the errors that could occur due to the quantization of $\rho^k_{start_l}$ and $\rho^k_{start_h}$ respectively and *int*(.) denotes the integer function.

Once these indices i.e., η^k_{sl} and η^k_{sh} are obtained, column C_k is scanned from η^k_{sl} to $\rho_{asize} - 1$ for n_{consec} non-zero entries. The value of n_{consec} depends on the density of the noise present and the value of $\Delta\rho$. Similarly C_k is also scanned from η^k_{sh} to 0 for n_{consec} non-zero entries. This operation is repeated for two different columns C_q and C_r.

After the columns are scanned, the extremities of the line segment are computed from eq 4.35 and eq 4.36. In eq 4.35 and eq 4.36 ρ^q_1 ,ρ^q_2, ρ^r_1 ,ρ^r_2, are determined from eq 4.32 where η^q_{fn} ,η^q_{ln}, η^r_{fn} ,η^r_{ln} are now replaced by η^q_{sl} ,η^q_{sh}, η^r_{sl} ,η^r_{sh} respectively. The columns C_q and C_r are selected by using the criteria mentioned in the previous sections i.e., $q > \theta_{peak}$ and $r > \theta_{peak}$ if $\theta_{predicted} < 45^\circ$ and $q < \theta_{peak}$ and $r < \theta_{peak}$ if $\theta_{predicted} \geq 45^\circ$. Results using this approach on synthetic images containing random noise are shown in fig 5.5.1 to fig 5.5.7. These are discussed elaborately in sec 5.5.

4.7 Summary

In this chapter, the limited number of approaches available in literature for obtaining the complete line segment description have been discussed. A new methodology for obtaining the complete line segment description from the spread in the accumulator array has been explained elaborately. The new approach has been developed for thin as well as thick line segments with and without using extrapolation. The new procedure has been refined effectively to overcome the drawbacks associated with the previous approaches by computing the extremities independent of $\theta_{predicted}$. A modification of the new approach to overcome the effect of noise has also been explained. Results obtained using this new approach and its modifications are presented and discussed in the next chapter.

5. RESULTS AND DISCUSSION

The various approaches to determine the endpoints of a straight line from the spread in the accumulator array were discussed in the previous chapter. The proposed methods have several advantages over the existing methods. The new methods can detect the endpoints of a thick line segment (which to our knowledge is the first attempt at determining the extremities of a thick line segment). Reasonably accurate values of the extremities are obtained by using large values of $\Delta\rho$ and $\Delta\theta$. This also reduces the memory space requirements and the computational complexity of the HT (this has been achieved, by abandoning the conventional way of computing ρ and θ accurately from the accumulator array and focussing on how to compute the extremities accurately; once this is done ρ and θ can be computed from the extremities).

The new methods have the disadvantage that the extremities of the line segments having discontinuities cannot be computed. And also the extremities of the line segments in images containing multiple lines cannot be computed using the proposed approach.

In this chapter, the results obtained using these approaches are discussed in detail. In sec 5.1 the results obtained by computing the length from the spread in the accumulator array (sec 4.4.1) are discussed, while in sec 5.2 results obtained by computing the length and extremities using $\theta_{predicted}$ (sec 4.4.2, sec

4.4.3 and sec 4.4.4) are presented. In sec 5.3 and sec 5.4, the results obtained by computing the length and extremities independent of $\theta_{predicted}$ (sec 4.5) are discussed. The results illustrating the noise performance (sec 4.6) are presented in sec 5.5.

All the results presented are in terms of number of pixels. Percentage error is not used since this could be misleading, because the error is independent of the length of the line. For the same value of the error in terms of pixels, the percentage error for two lines having different lengths would be different. Hence the error is computed in terms of pixels.

5.1 Computation of length from the spread in the accumulator array

In this approach, the expression developed by Van veen & Groen [25], for spread in the accumulator array forms the basis. The expression for the length of the line segment is given by eq 4.16. This expression was applied to various synthetic images. Table 5.1.1 and Table 5.1.2 give some of the results obtained using this approach. The accuracy of l_{ave} depends on the number of columns over which eq 4.16 is averaged i.e., the value of M. From the results it can be seen that, if $\Delta\rho$ is made small (which is done for improving the accuracy of the computed value of ρ), the accuracy in the computed value of the length

Table 5.1.1

Simulation results for computing the length of the line from the spread in the accumulator array.
$N = 128$, $\theta_{actual} = 23^{\circ}$, $\rho_{actual} = 90$, $n_f = 101$, $l_{actual} = 105.7$

Number of bars on each side of the peak M	$\Delta\rho = 0.166$ $\Delta\theta = 0.9$ $\rho_{predicted} = 89.665$ $\theta_{predicted} = 23.399$ l_{ave}	$\Delta\rho = 0.083$ $\Delta\theta = 0.9$ $\rho_{predicted} = 89.624$ $\theta_{predicted} = 23.399$ l_{ave}	$\Delta\rho = 0.074$ $\Delta\theta = 0.4$ $\rho_{predicted} = 89.899$ $\theta_{predicted} = 23.199$ l_{ave}	$\Delta\rho = 0.037$ $\Delta\theta = 0.4$ $\rho_{predicted} = 89.389$ $\theta_{predicted} = 23.199$ l_{ave}
1	101.869	93.130	123.316	84.560
2	102.580	95.071	121.909	94.603
3	102.022	95.525	114.999	86.979
4	95.851	83.953	114.146	86.818
5	87.807	75.127	113.230	84.255

Table 5.1.2

Simulation results for computing the length of the line from the spread in the accumulator array.
$N = 128$, $\theta_{actual} = 73^{\circ}$, $\rho_{actual} = 90$, $n_f = 81$, $l_{actual} = 83.8$

Number of bars on each side of the peak M	$\Delta\rho = 0.132$ $\Delta\theta = 0.9$ $\rho_{predicted} = 89.171$ $\theta_{predicted} = 72.899$ l_{ave}	$\Delta\rho = 0.066$ $\Delta\theta = 0.9$ $\rho_{predicted} = 89.269$ $\theta_{predicted} = 72.899$ l_{ave}	$\Delta\rho = 0.102$ $\Delta\theta = 0.7$ $\rho_{predicted} = 89.856$ $\theta_{predicted} = 72.599$ l_{ave}	$\Delta\rho = 0.051$ $\Delta\theta = 0.7$ $\rho_{predicted} = 89.144$ $\theta_{predicted} = 73.256$ l_{ave}
1	89.364	75.429	105.796	70.995
2	84.903	75.016	96.335	68.492
3	82.943	71.770	92.071	69.133
4	80.373	63.846	89.247	68.562
5	78.938	61.374	83.859	62.829

decreases. This happens, because as $\Delta\rho$ is made smaller, the phenomenon of zero votes between non-zero votes as explained in sec 4.5.1 occurs, thereby leading to inaccurate values of n_ρ^k. Another drawback in using this approach is that, only the length and the normal parameters can be estimated. The extremities of the line segment cannot be determined using the approach discussed in sec 4.4.1, which means that the objective of obtaining the complete line segment description from the spread in the accumulator array remains incomplete. Techniques to overcome this shortcoming were discussed in sec 4.4.2 to sec 4.5. Results from these approaches are discussed in the subsequent sections.

5.2 Length and extremities using $\theta_{predicted}$

The expressions to determine the extremities of the line segment from the spread in the accumulator array were derived in sec 4.4.2 (eq 4.23) and sec 4.4.3 (eq 4.31). The approach in sec 4.4.2 does not involve the use of extrapolation while that in sec 4.4.3 uses extrapolation in determining the extremities of the line segment. Once the extremities of a line segment are determined, the length could be computed from eq 4.24. The results of these two approaches are discussed in the following two sub sections.

5.2.1 Length and extremities without using extrapolation

This section presents the results obtained using the techniques discussed in sec 4.4.2 for determining the extremities and length of the line segment. The results are shown in fig 5.2.1.1 to fig 5.2.1.3. These plots show the variation in error in the x-co-ordinates, the y-co-ordinates, and the computed length respectively, with $d_{k,\theta_{peak}}$ where C_k is the column in the accumulator array whose bars are intersected with the predicted line segment. The error is in terms of pixels and is shown for two different values of $\Delta\theta$. The point '0' on the x-axis corresponds to $d_{k,\theta_{peak}} = 0$. The points on the right side of the '0' correspond to $d_{k,\theta_{peak}}$, $k > \theta_{peak}$ and the points on the left side of '0' correspond to $d_{k,\theta_{peak}}$, $k < \theta_{peak}$

It can be concluded from these plots, that as $d_{k,\theta_{peak}}$ increases, the error (in terms of pixels) decreases. This is due to the fact, that the calculation of ρ_1 and ρ_2 (eq 4.17 and eq 4.18 respectively) using C_k, where $|k - \theta_{peak}|$ is small, can be rather inaccurate. Since near the peak, the spread in a column is such that, it could result in low votes in the first and last non-zero cells in that column. The vote count in the first and last non-zero cells in a column could be as low as 1 or 2, unlike the vote count in the other cells of the column which is relatively higher. This could result in inaccurate calculation of ρ_1 and ρ_2, thereby leading to rather high errors in the computed extremities and the length using the

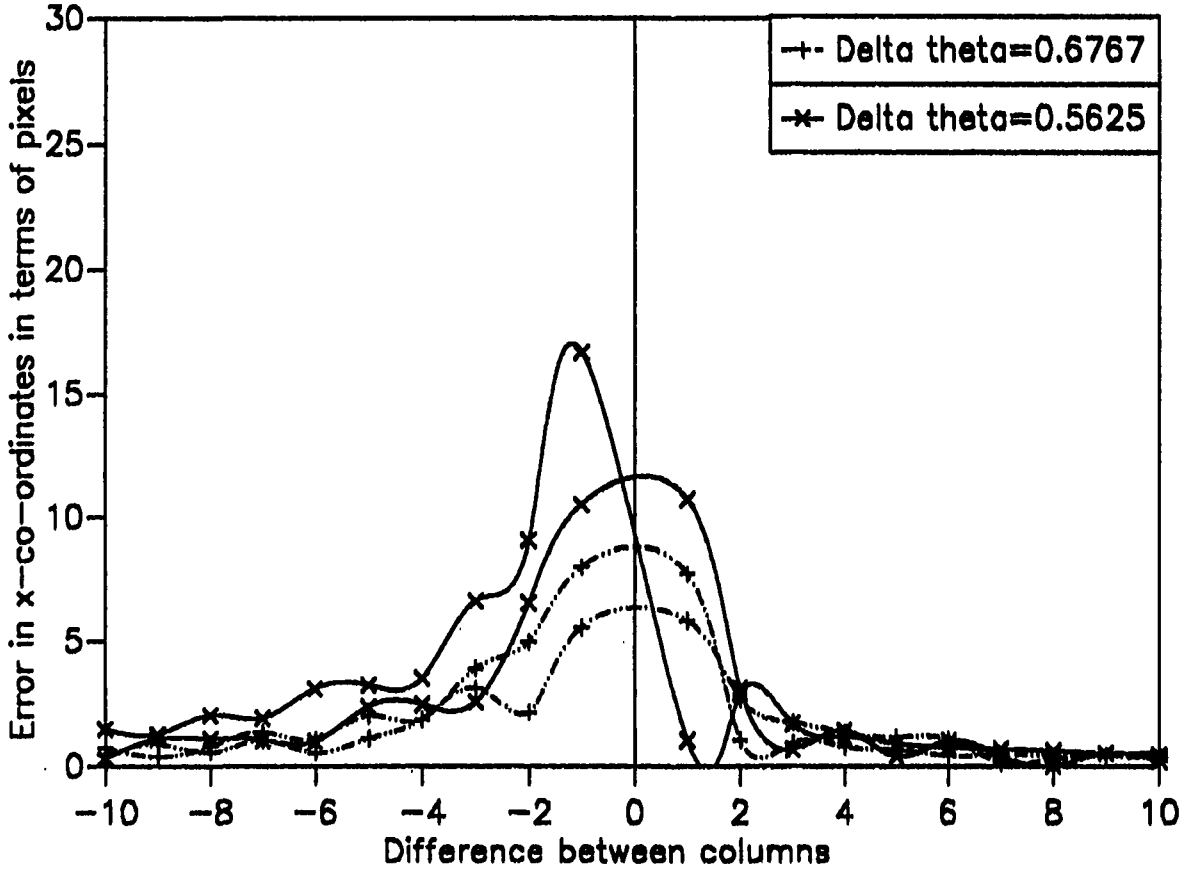


Fig 5.2.1.1 Variation of error in the x-co-ordinates vs $d_{j,\theta_{peak}}$, N = 128, θ_{actual} = 15°, ρ_{actual} = 90, l_{actual} = 70.604 With $\Delta\rho$ = 0.299 the parameters computed from the peak are : for $\Delta\theta$ = 0.5625 --- $\rho_{predicted}$ = 89.054, $\theta_{predicted}$ = 14.625 for $\Delta\theta$ = 0.6767 --- $\rho_{predicted}$ = 89.353, $\theta_{predicted}$ = 14.887

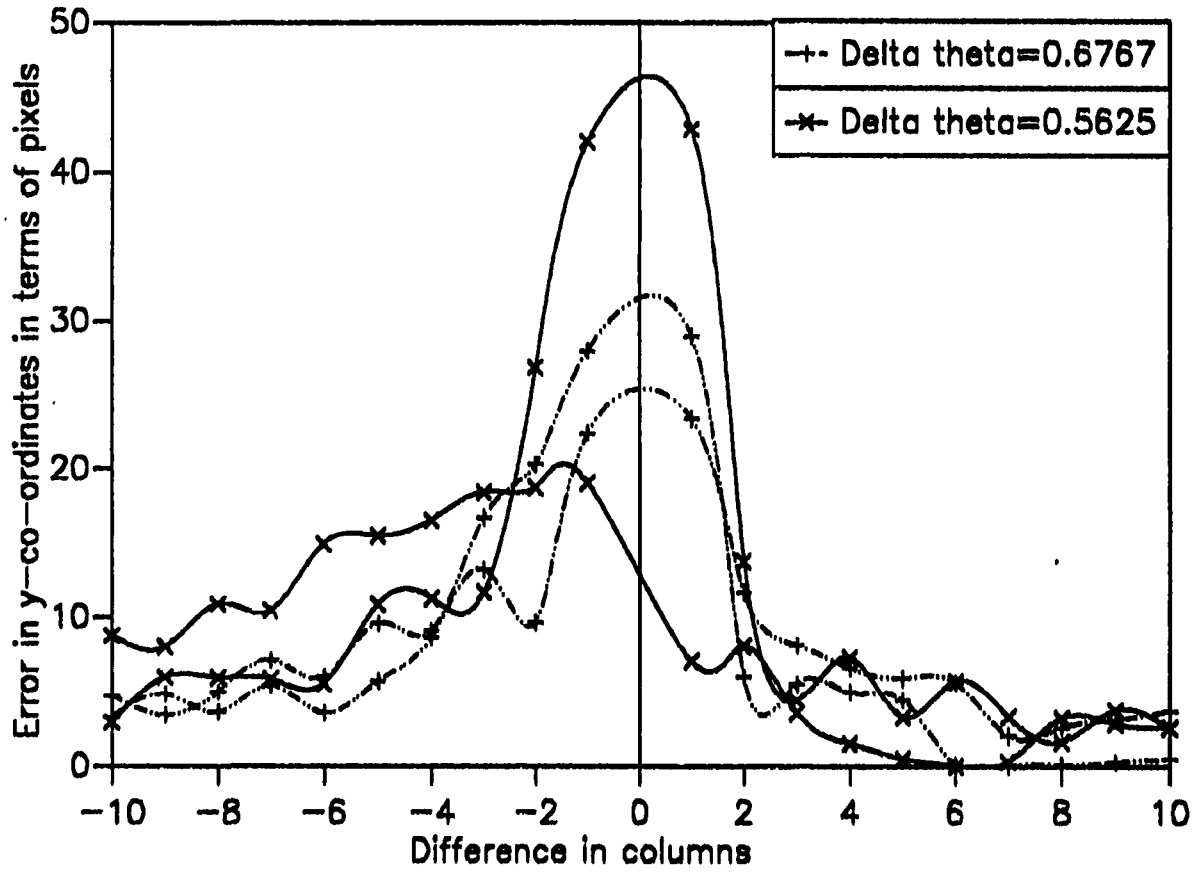


Fig 5.2.1.2 Variation of error in the y-co-ordinates vs $d_{j,\theta_{peak}}$. N = 128, θ_{actual} = 15°, ρ_{actual} = 90, l_{actual} = 70.604 With $\Delta\rho$ = 0.299 the parameters computed from the peak are : for $\Delta\theta$ = 0.5625 --- $\rho_{predicted}$ = 89.054, $\theta_{predicted}$ = 14.625 for $\Delta\theta$ = 0.6767 --- $\rho_{predicted}$ = 89.353, $\theta_{predicted}$ = 14.887

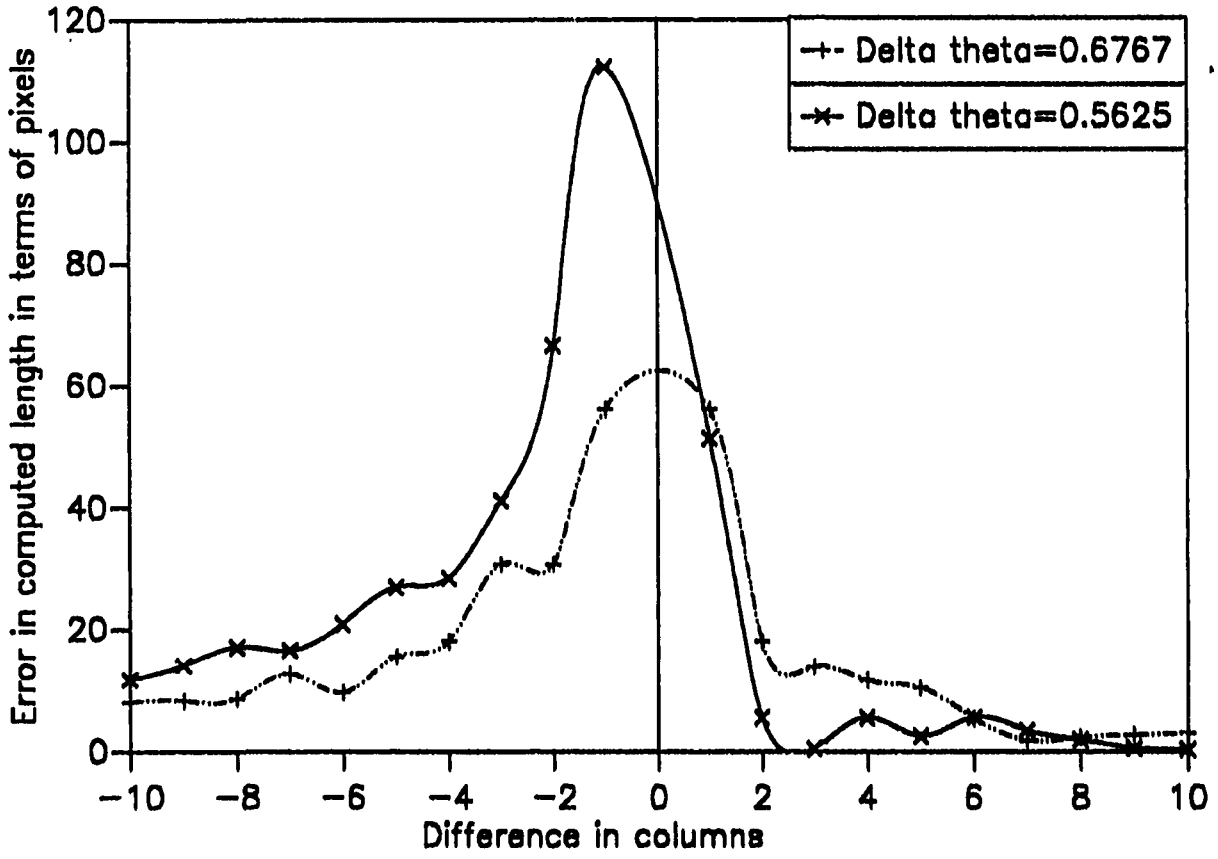


Fig 5.2.1.3 Variation of error in $l_{computed}$ vs $d_{j,\theta_{peak}}$. N = 128, θ_{actual} = 15°, ρ_{actual} = 90, l_{actual} = 70.604 With $\Delta\rho$ = 0.299 the parameters computed from the peak are : for $\Delta\theta$ = 0.5625 --- $\rho_{predicted}$ = 89.054, $\theta_{predicted}$ = 14.625 for $\Delta\theta$ = 0.6767 --- $\rho_{predicted}$ = 89.353, $\theta_{predicted}$ = 14.887

columns near the peak. As $d_{k,\theta_{peak}}$ increases, the spread becomes more uniform and the corresponding errors are less.

5.2.2 Length and extremities using extrapolation

As mentioned in sec 4.4.2 the co-ordinates of the endpoints of a line segment are computed by determining the point of intersection between the predicted line segment and $b_{\theta_{low},k}$ or $b_{\theta_{high},k}$. While doing this it is not necessary that an endpoint of the line segment lies exactly on the boundaries of the bar that is used in determining that endpoint. This error could be further reduced by using extrapolation techniques discussed in sec 4.4.3. The results for this approach are shown in fig 5.2.2.1 to fig 5.2.2.9. Fig 5.2.2.1 to fig 5.2.2.3 show the variation of error in the x-co-ordinates, the y-co-ordinates and the computed length for $\Delta\rho = 0.2245$ and $\Delta\theta = 0.9$, while fig 5.2.2.4 to fig 5.2.2.6 show these same errors for $\Delta\rho = 0.2990$ and $\Delta\theta = 0.9$. In sec 4.4.3, three different types of extrapolations were discussed. The results for these three types of extrapolations are similar for those values of $\Delta\rho$, for which the phenomenon of zero votes between non-zero votes is not pronounced. Again for extrapolation also, the error in computed co-ordinates decreases as the angle between the predicted line segment and $b_{\theta_{low},k}$ or $b_{\theta_{high},k}$ increases. Fig 5.2.2.7 to Fig 5.2.2.9 show the variation

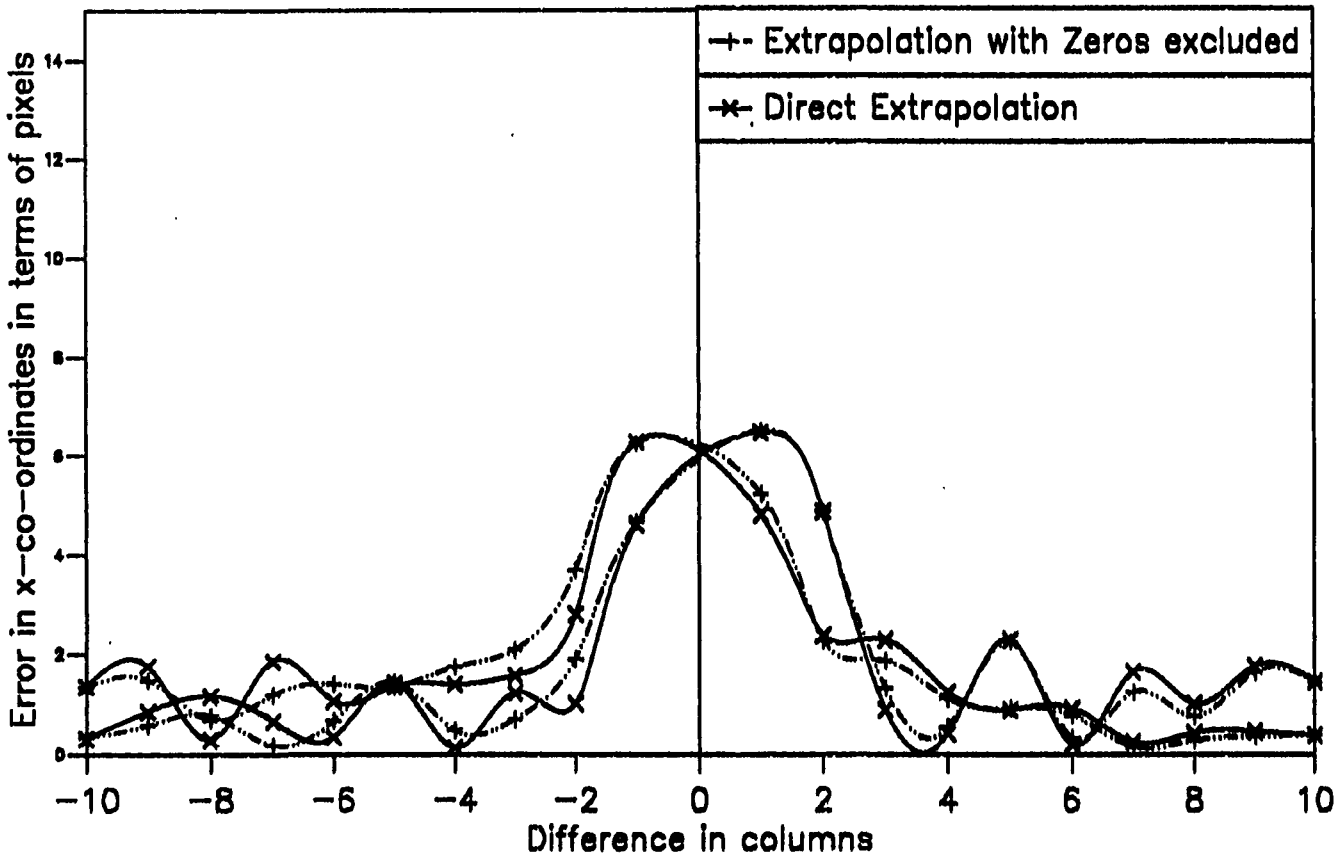


Fig 5.2.2.1 Variation of error in the x-co-ordinates vs $d_{j,\theta_{peak}}$, when extrapolation is used $N = 128$, $\theta_{actual} = 45^{\circ}$, $\rho_{actual} = 90$, $l_{actual} = 89.095$ For $\Delta\rho = 0.2245$ $\Delta\theta = 0.9$, the parameters computed from the peak are $\rho_{predicted} = 89.095$, $\theta_{predicted} = 44.999$

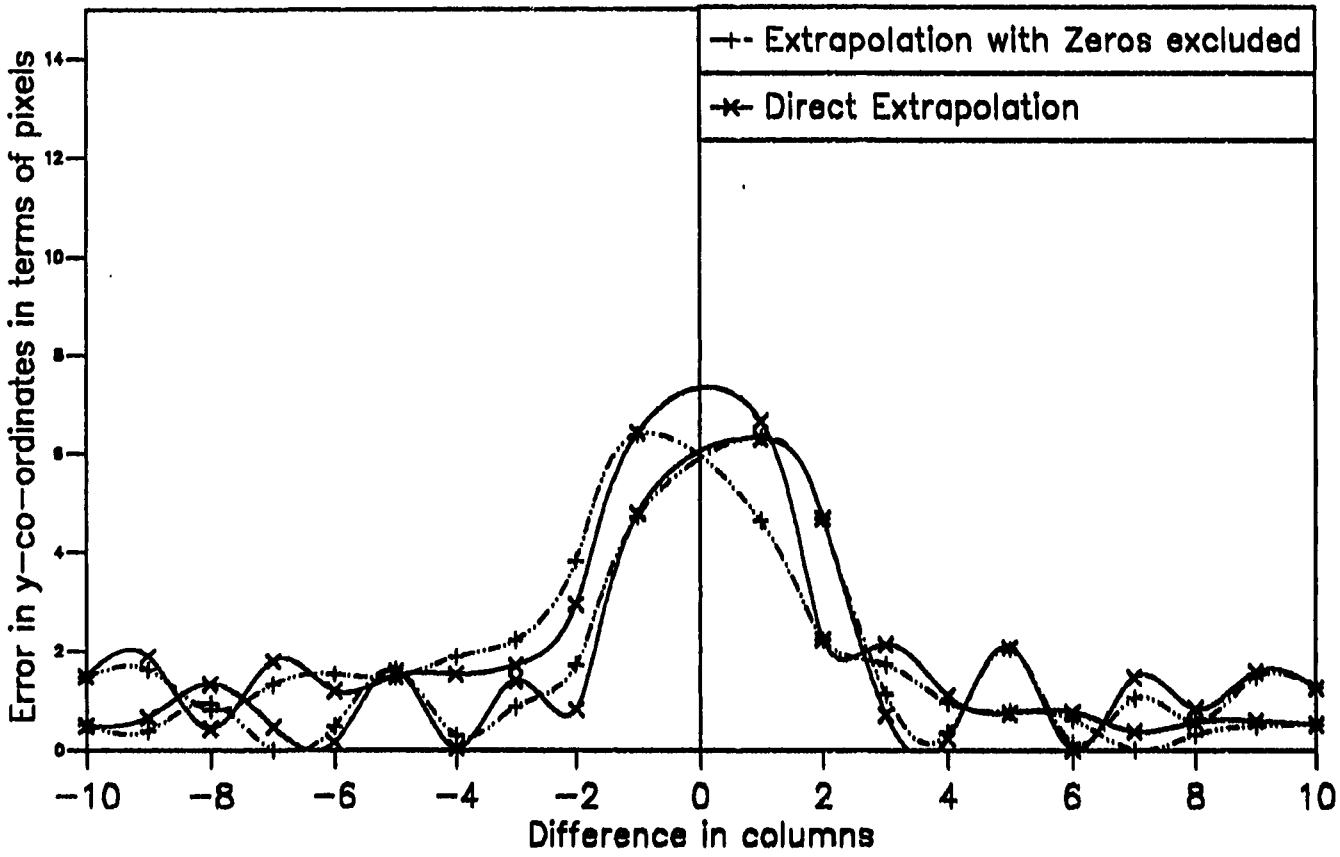


Fig 5.2.2.2 Variation of error in the y-co-ordinates vs $d_{\rho,\theta_{peak}}$, when extrapolation is used $N = 128$, $\theta_{actual} = 45^{\circ}$, $\rho_{actual} = 90$, $I_{actual} = 89.095$ For $\Delta\rho = 0.2245$ $\Delta\theta = 0.9$, the parameters computed from the peak are $\rho_{predicted} = 89.095$, $\theta_{predicted} = 44.999$

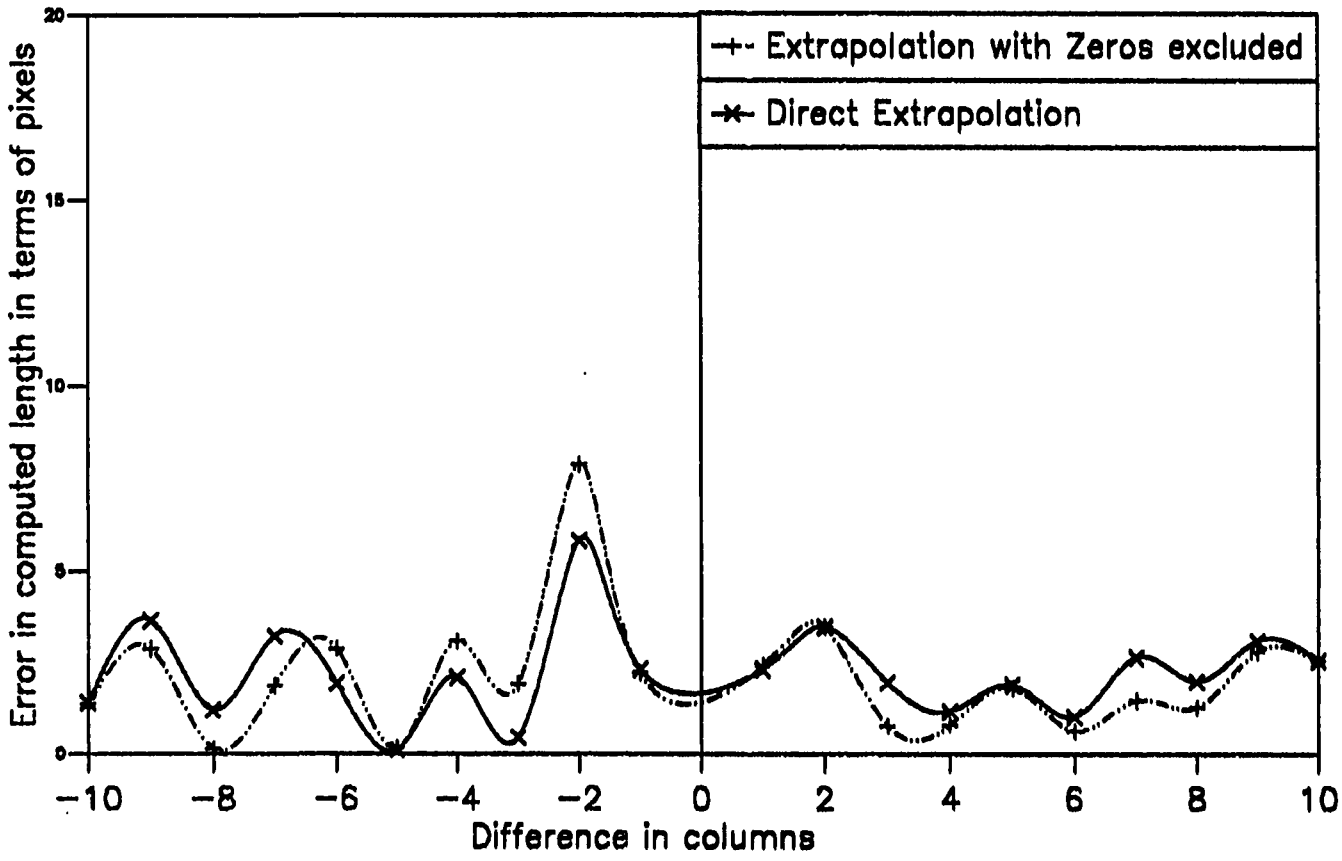


Fig 5.2.2.3 Variation of error in $l_{computed}$ vs $d_{j,\theta_{peak}}$, when extrapolation is used $N = 128$, $\theta_{actual} = 45^{\circ}$, $\rho_{actual} = 90$, $l_{actual} = 89.095$ For $\Delta\rho = 0.2245$ $\Delta\theta = 0.9$, the parameters computed from the peak are $\rho_{predicted} = 89.095$, $\theta_{predicted} = 44.999$

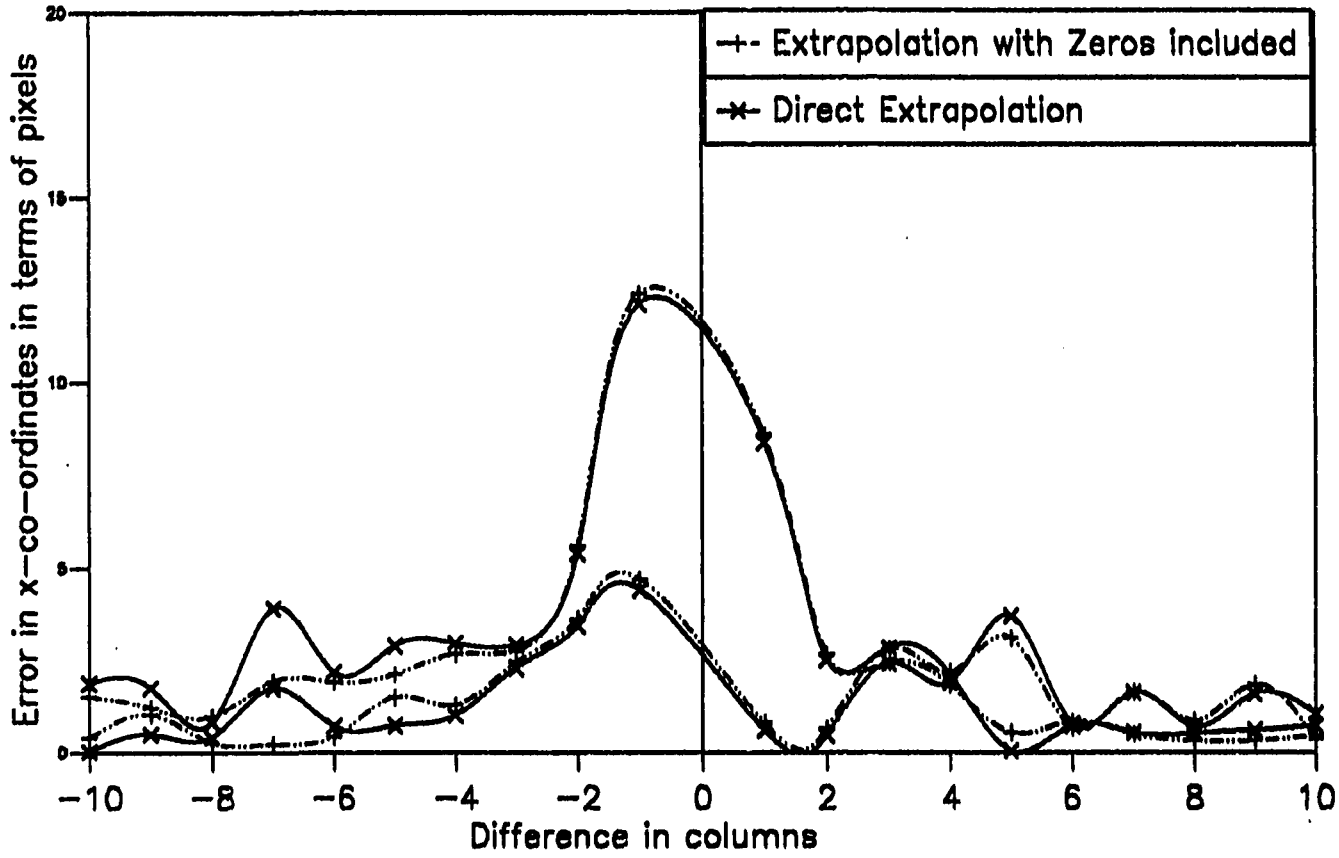


Fig 5.2.2.4 Variation of error in the x-co-ordinates vs $d_{f,\theta_{peak}}$, when extrapolation is used $N = 128$, $\theta_{actual} = 45^{\circ}$, $\rho_{actual} = 90$, $l_{actual} = 66.468$ For $\Delta\rho = 0.299$ $\Delta\theta = 0.9$, the parameters computed from the peak are $\rho_{predicted} = 89.652$, $\theta_{predicted} = 44.999$

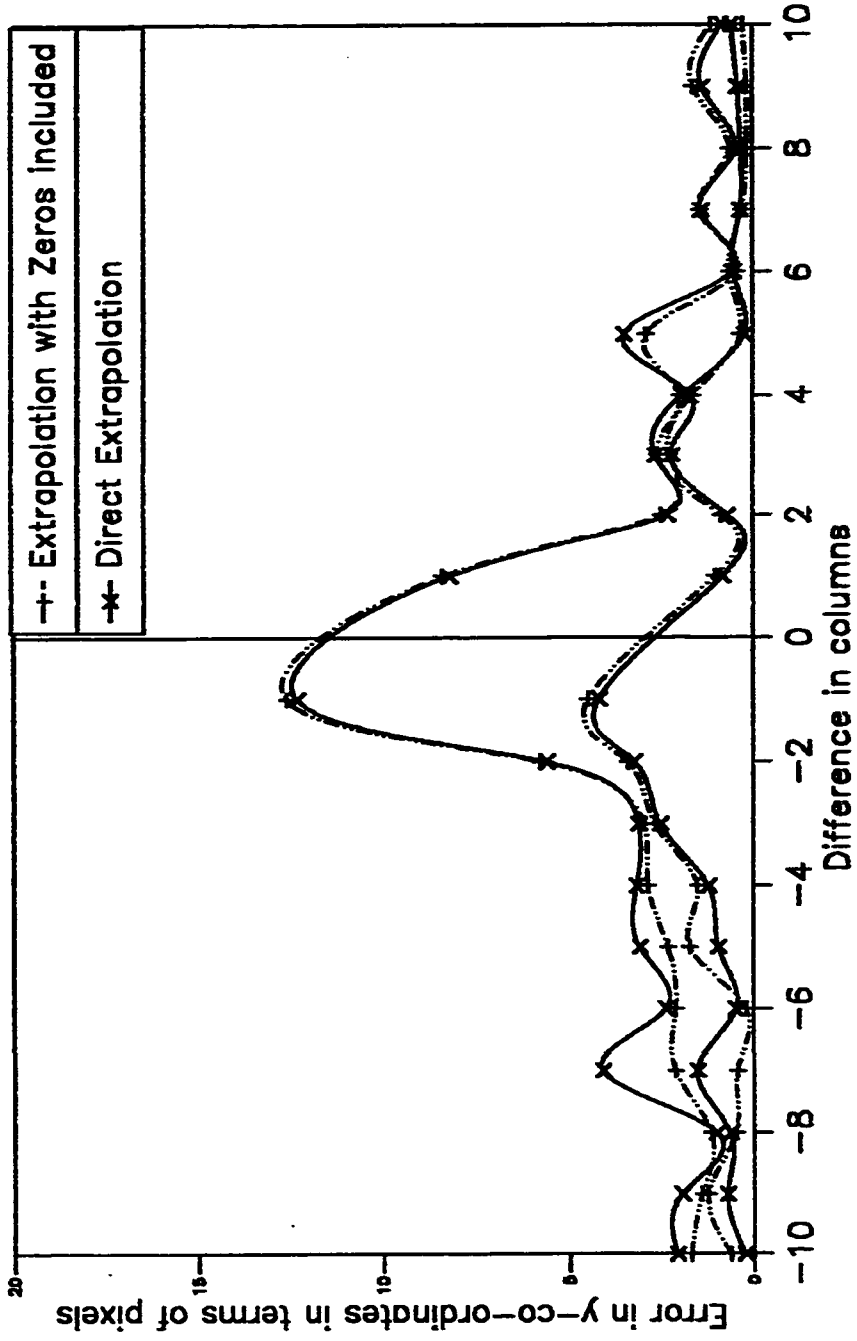


Fig 5.2.2.5 Variation of error in the y-co-ordinates vs $d_{l,\theta_{peak}}$ when extrapolation is used $N = 128$, $\theta_{actual} = 45°$, $\rho_{actual} = 90$, $l_{actual} = 66.468$ For $\Delta\rho = 0.299$ $\Delta\theta = 0.9$, the parameters computed from the peak are $\rho_{predicted} = 89.652$, $\theta_{predicted} = 44.999$

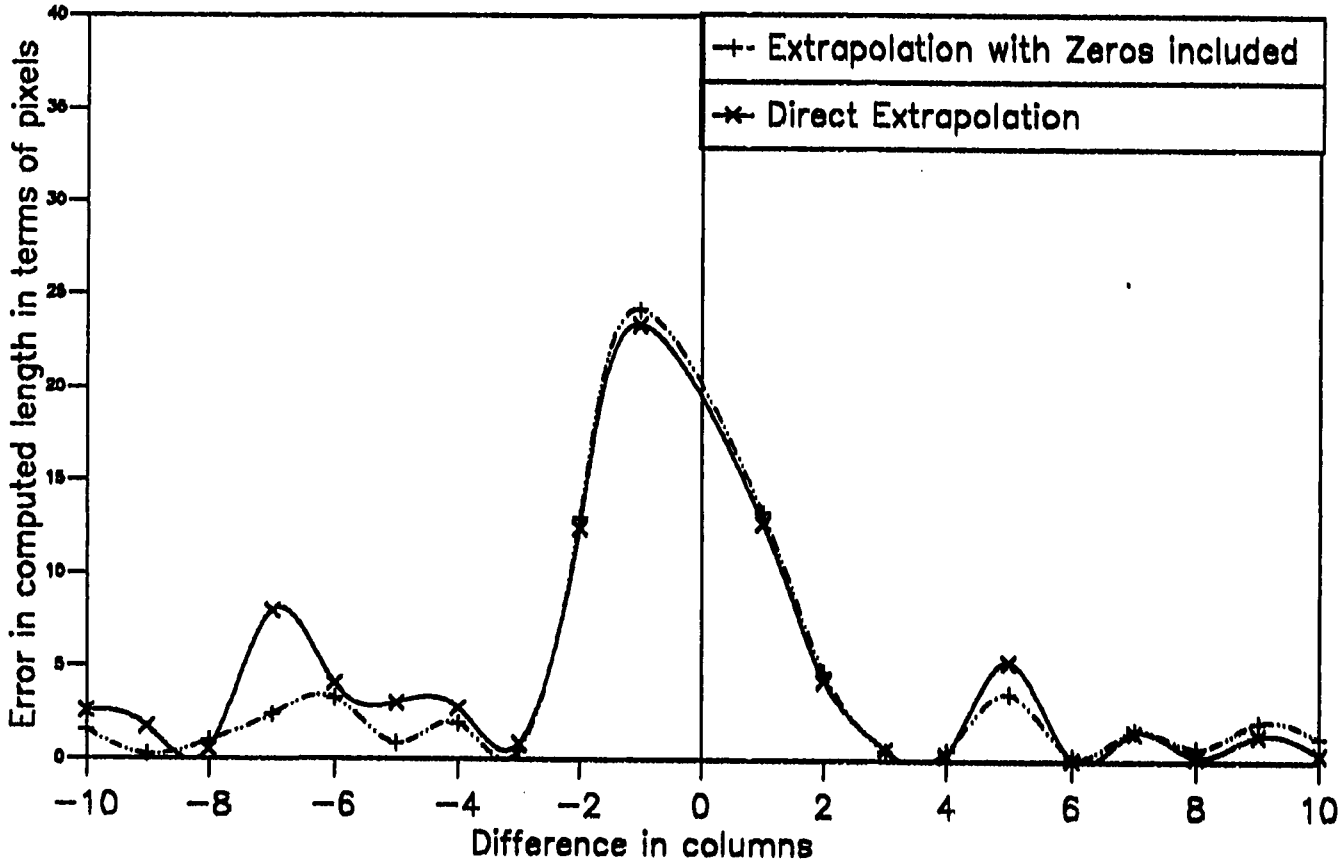


Fig 5.2.2.6 Variation of error in $l_{computed}$ vs $d_{\rho,\theta_{peak}}$ when extrapolation is used $N = 128$, $\theta_{actual} = 45^{\circ}$, $\rho_{actual} = 90$, $l_{actual} = 66.468$ For $\Delta\rho = 0.299$ $\Delta\theta = 0.9$, the parameters computed from the peak are $\rho_{predicted} = 89.652$, $\theta_{predicted} = 44.999$

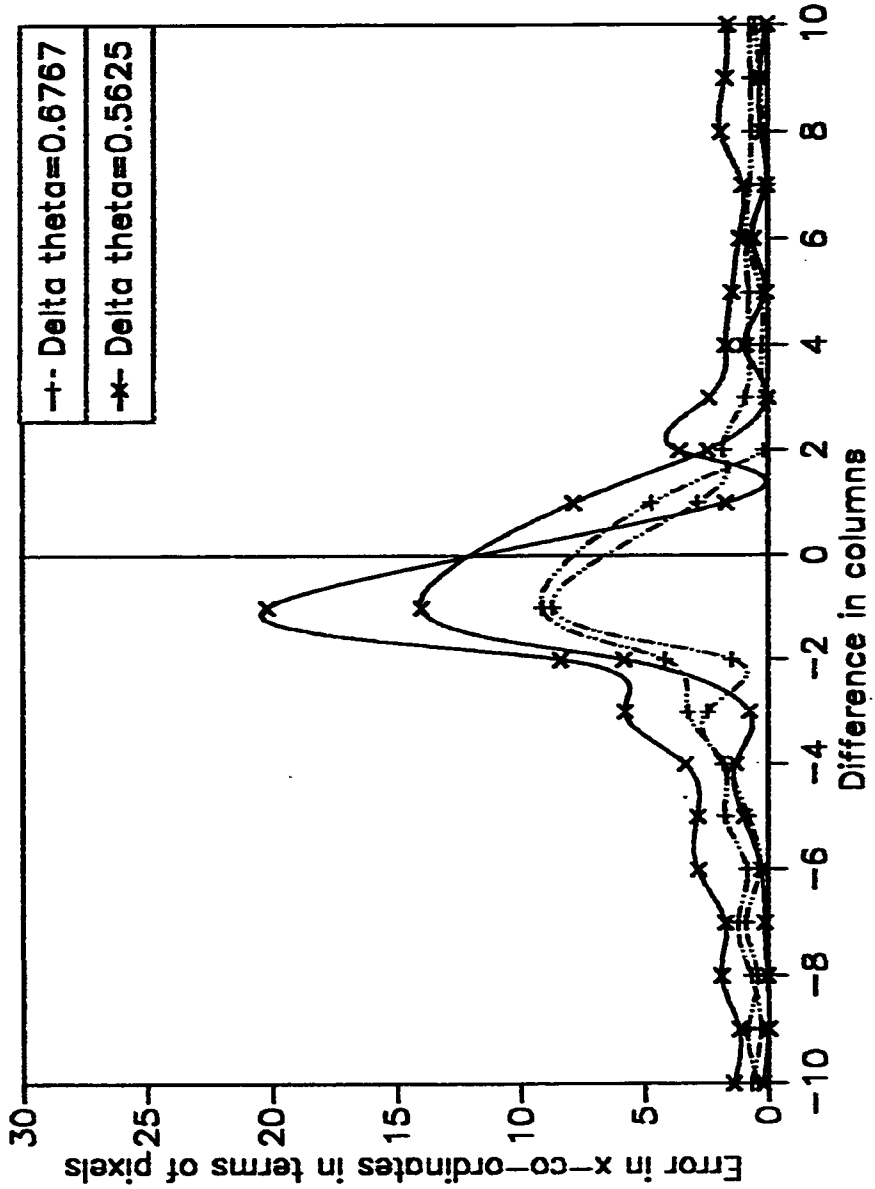


Fig 5.2.2.7 Variation of error in the x-co-ordinates vs $d_{j,\theta_{peak}}$ when extrapolation with zeros excluded is used $N = 128$, $\theta_{actual} = 15'$, $\rho_{actual} = 90$, $l_{actual} = 70.604$ With $\Delta\rho = 0.299$ the parameters computed from the peak are : for $\Delta\theta = 0.5625$ --- $\rho_{predicted} = 89.054$, $\theta_{predicted} = 14.625$ for $\Delta\theta = 0.6767$ --- $\rho_{predicted} = 89.353$, $\theta_{predicted} = 14.887$

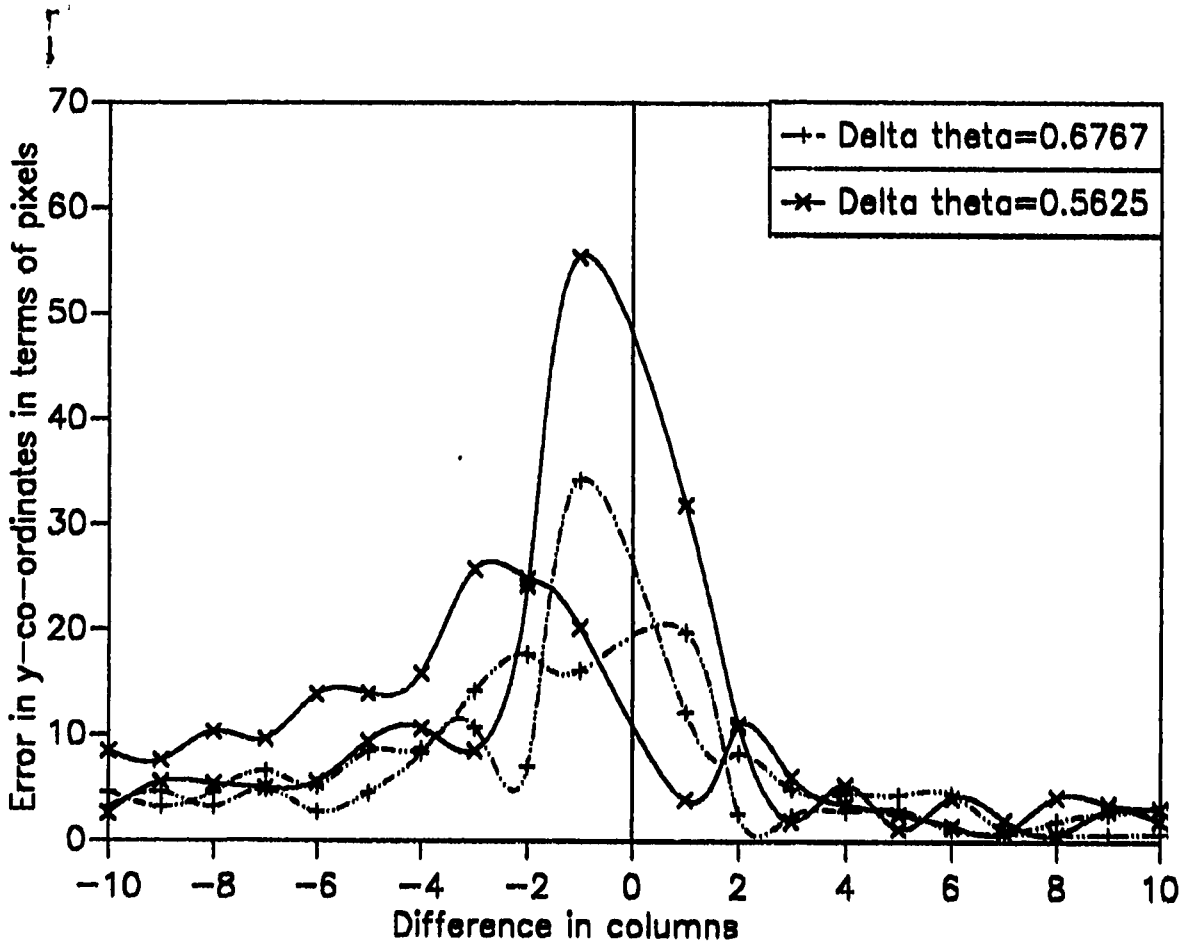


Fig 5.2.2.8 Variation of error in the y-co-ordinates vs $d_{j,\rho_{peak}}$ when extrapolation with zeros excluded is used $N = 128$, $\theta_{actual} = 15^{\circ}$, $\rho_{actual} = 90$, $l_{actual} = 70.604$ With $\Delta\rho = 0.299$ the parameters computed from the peak are : for $\Delta\theta = 0.5625$ --- $\rho_{predicted} = 89.054$, $\theta_{predicted} = 14.625$ for $\Delta\theta = 0.6767$ --- $\rho_{predicted} = 89.353$, $\theta_{predicted} = 14.887$

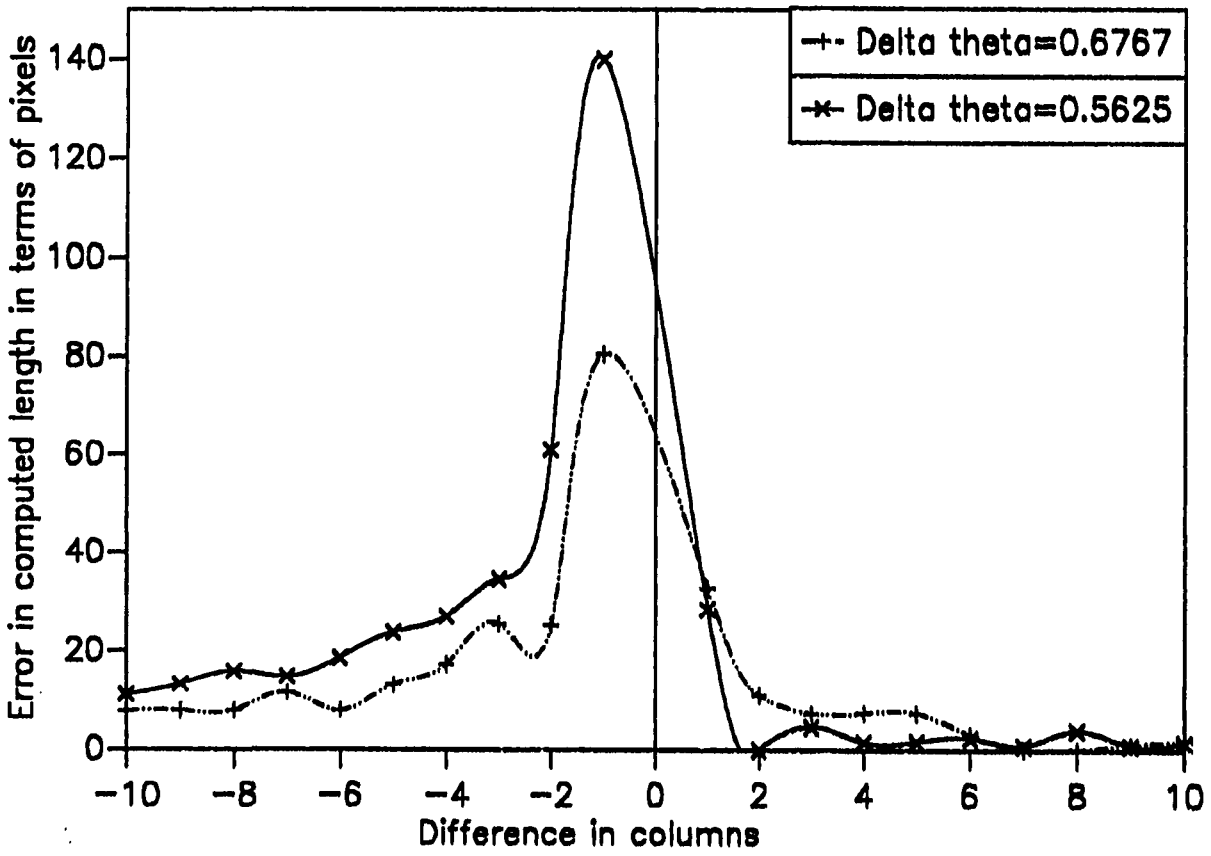


Fig 5.2.2.9 Variation of error in $l_{computed}$ vs $d_{f,\theta_{peak}}$, when extrapolation with zeros excluded is used N = 128, 0_{actual} = 15°, ρ_{actual} = 90, l_{actual} = 70.604 With $\Delta\rho$ = 0.299 the parameters computed from the peak are : for $\Delta 0$ = 0.5625 --- $\rho_{predicted}$ = 89.054, $0_{predicted}$ = 14.625 for $\Delta 0$ = 0.6767 --- $\rho_{predicted}$ = 89.353, $0_{predicted}$ = 14.887

of error in the x-co-ordinates, the y-co-ordinates and the computed length for different values of $\Delta\theta$. For fixed $d_{k,\theta_{peak}}$ the errors for small values of $\Delta\theta$ are higher than errors for large values of $\Delta\theta$ as shown in fig 5.2.2.7 to fig 5.2.2.9. This is because, as $\Delta\theta$ is made small for a fixed value of $d_{k,\theta_{peak}}$, the angle between $b_{\theta_{peak}-k}$ or $b_{\theta_{peak}+k}$ and the predicted line segment decreases, which results in a decrease in the accuracy of the computed co-ordinates of the endpoints. Hence, using higher values of $\Delta\theta$ would result in more accurately computed extremities of the line segment. Higher values of $\Delta\theta$ would also result in reduction in the computational complexity of the HT.

With the approach discussed in sec 4.2.2 and sec 4.2.3, it is possible that the intersection of the predicted line segment and $b_{\theta_{peak}-k}$ or $b_{\theta_{peak}+k}$ may actually lie outside the image plane. This phenomenon happens when the bars in C_k are chosen such that k is in the vicinity of θ_{peak}. This will lead to the computation of those points as extremities, which are not even the feature points of the image. This serious shortcoming can be avoided by making the computation of endpoints independent of the predicted line segment as was discussed in sec 4.5. Results using this method are presented in the next section.

5.3 Complete Line Segment Description without using $\theta_{predicted}$

It is desirable that, the endpoints are computed independent of the predicted line segment. This ensures that the errors occurring in $\rho_{predicted}$ and $\theta_{predicted}$ do not reappear in the computation of the co-ordinates of the endpoints. This can be achieved as discussed in sec 4.5 and the corresponding expressions for the co-ordinates of the extremities of a line segment are given by eq 4.35 and eq 4.36. Once the extremities are detected accurately, $\rho_{computed}$ and $\theta_{computed}$ can be computed accurately using eq 4.41 and eq 4.40 respectively. Eq 4.35 and eq 4.36 were used in determining the extremities of line segments for different values of θ_{actual} and from these computed extremities the length was calculated using eq 4.24. The error in the computed value of length is plotted against $d_{q,r}$, in fig 5.3.1 to fig 5.3.5, for different line segments.

From these graphs, it is seen that as the difference between the columns increases (which means that the angle between the normals to the bars in the two columns increases), the error decreases. The maximum error that could be present in the computed x and y co-ordinates was derived in eq 4.37 and eq 4.38 respectively. These expressions are corroborated by the results obtained which are shown in fig 5.3.1 to fig 5.3.5. Theoretically, from eq 4.37 and eq 4.38, as the difference $(\theta_r - \theta_q)$ increases, the sine of the difference also increases, resulting in a decrease in the error in the computed values of the x and y co-

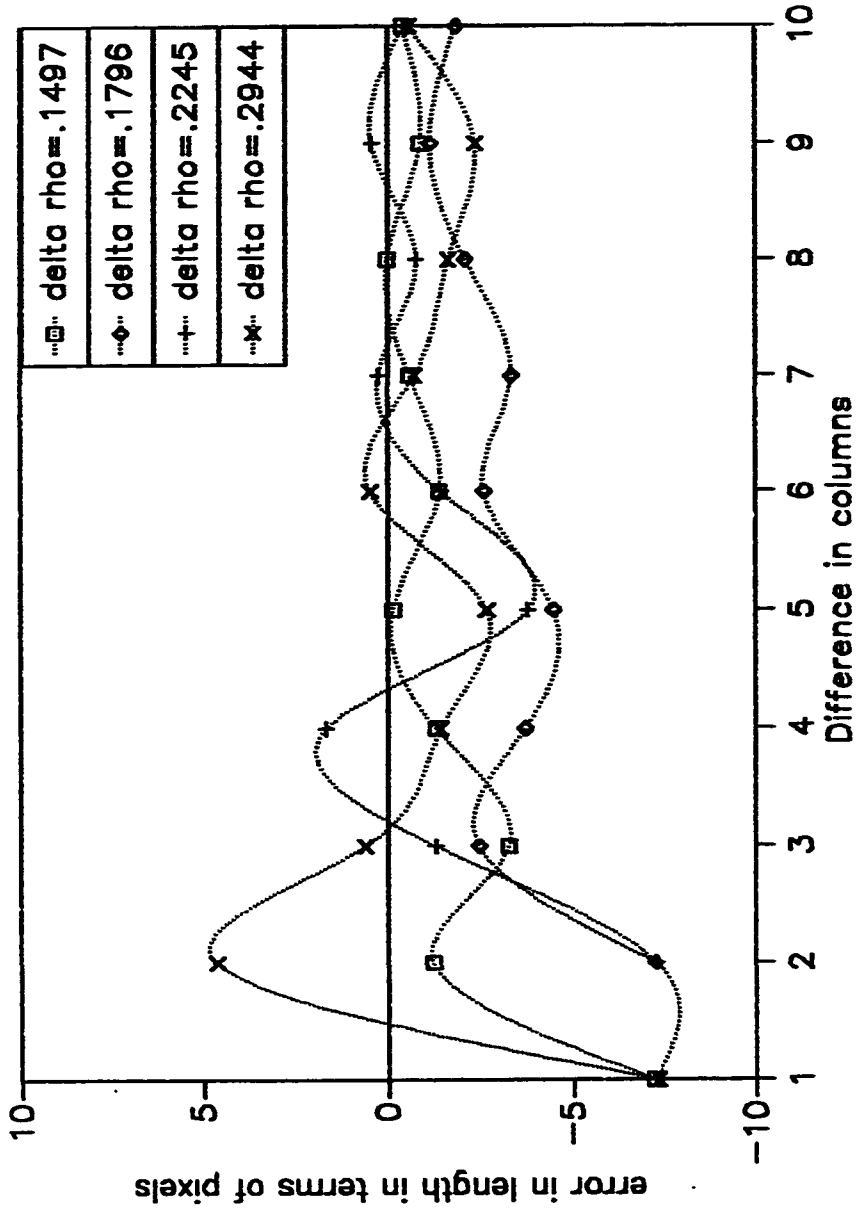


Fig 5.3.1.a Variation in error in $l_{computed}$ vs $d_{q,r}$ for $\Delta\theta = 0.7087$ and different $\Delta\rho$. $N = 128$, $\theta_{actual} = 15^{\circ}$, $\rho_{actual} = 90$.

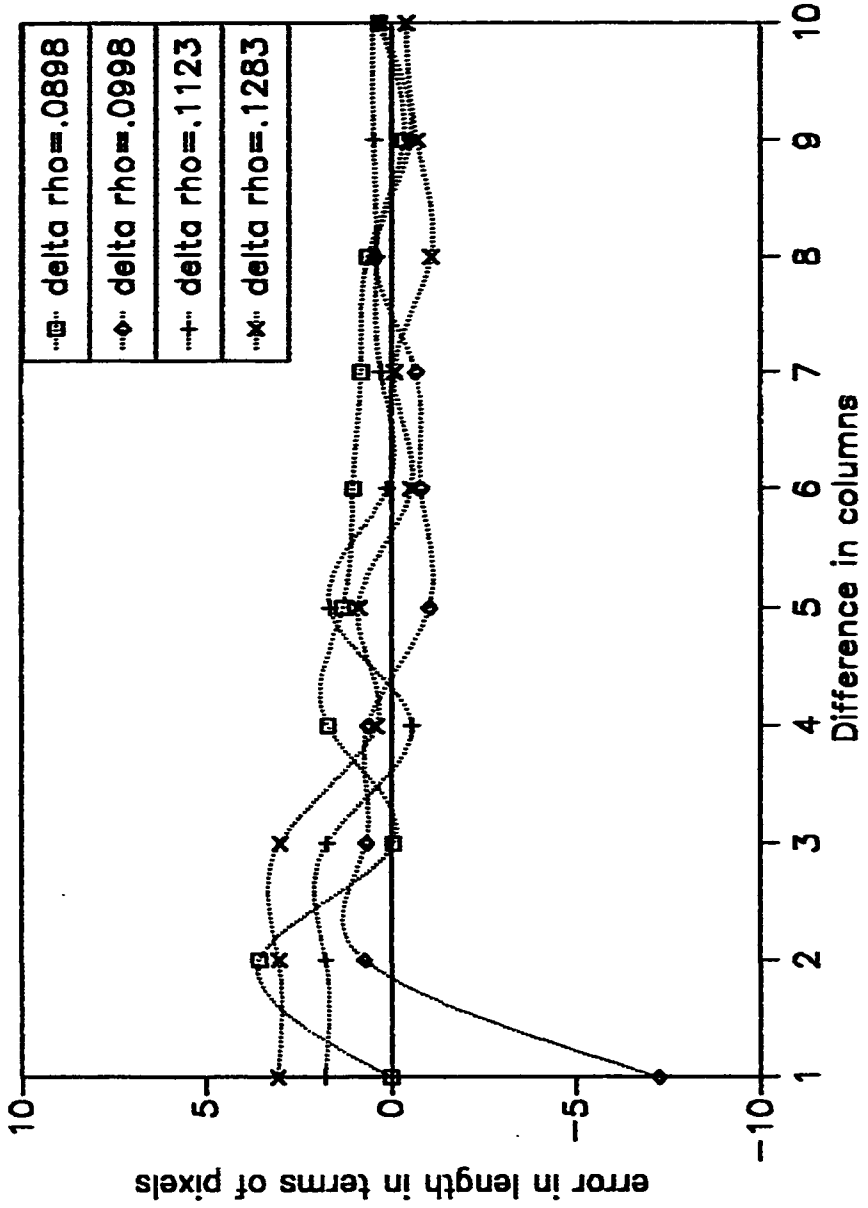


Fig 5.3.1.b Variation in error in $l_{computed}$ vs $d_{x,y}$ for $\Delta\theta = 0.7087$ and different $\Delta\rho$. $N = 128$. $\theta_{actual} = 15°$, $\rho_{actual} = 90$.

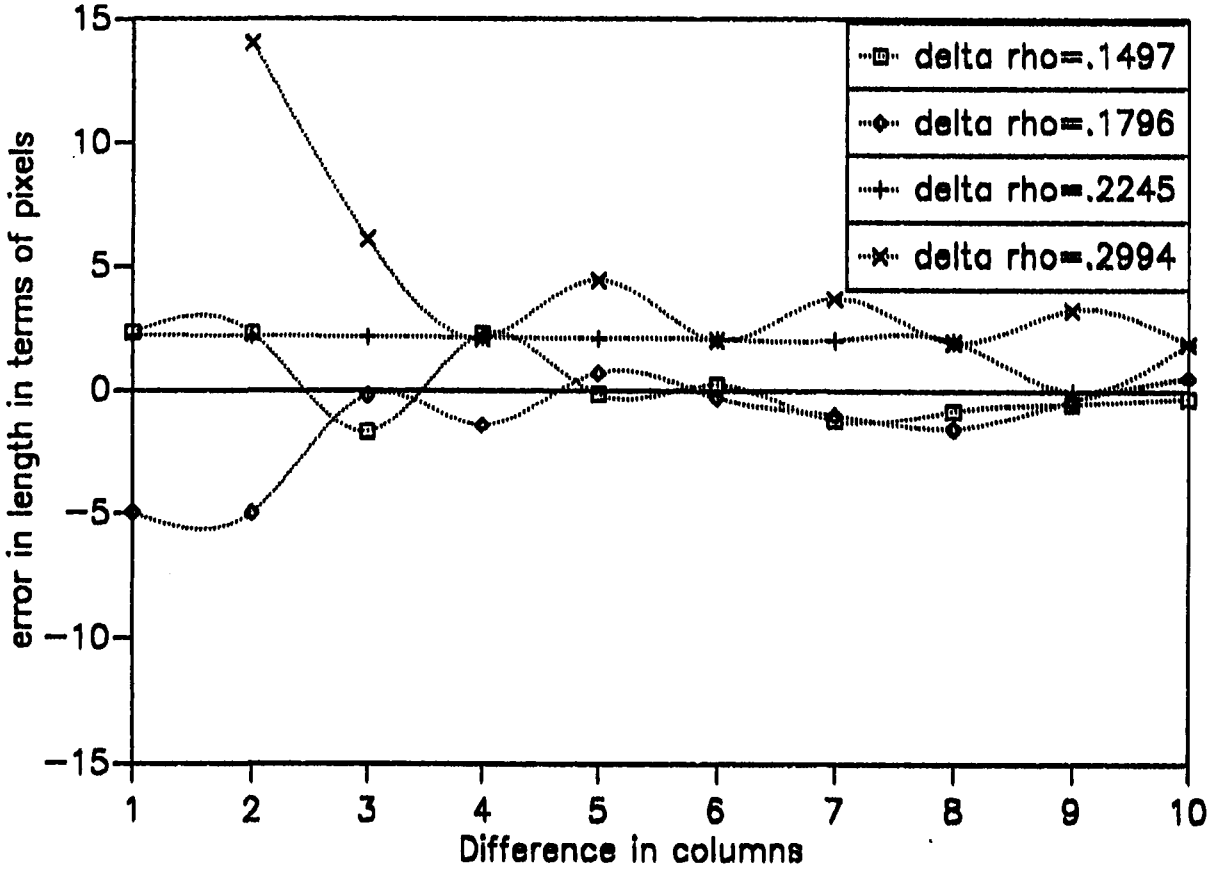


Fig 5.3.2.a Variation in error in $l_{computed}$ vs $d_{q,r}$ for $\Delta\theta = 0.7087$ and different $\Delta\rho$. $N = 128$, $\theta_{actual} = 25^{\circ}$, $\rho_{actual} = 90$.

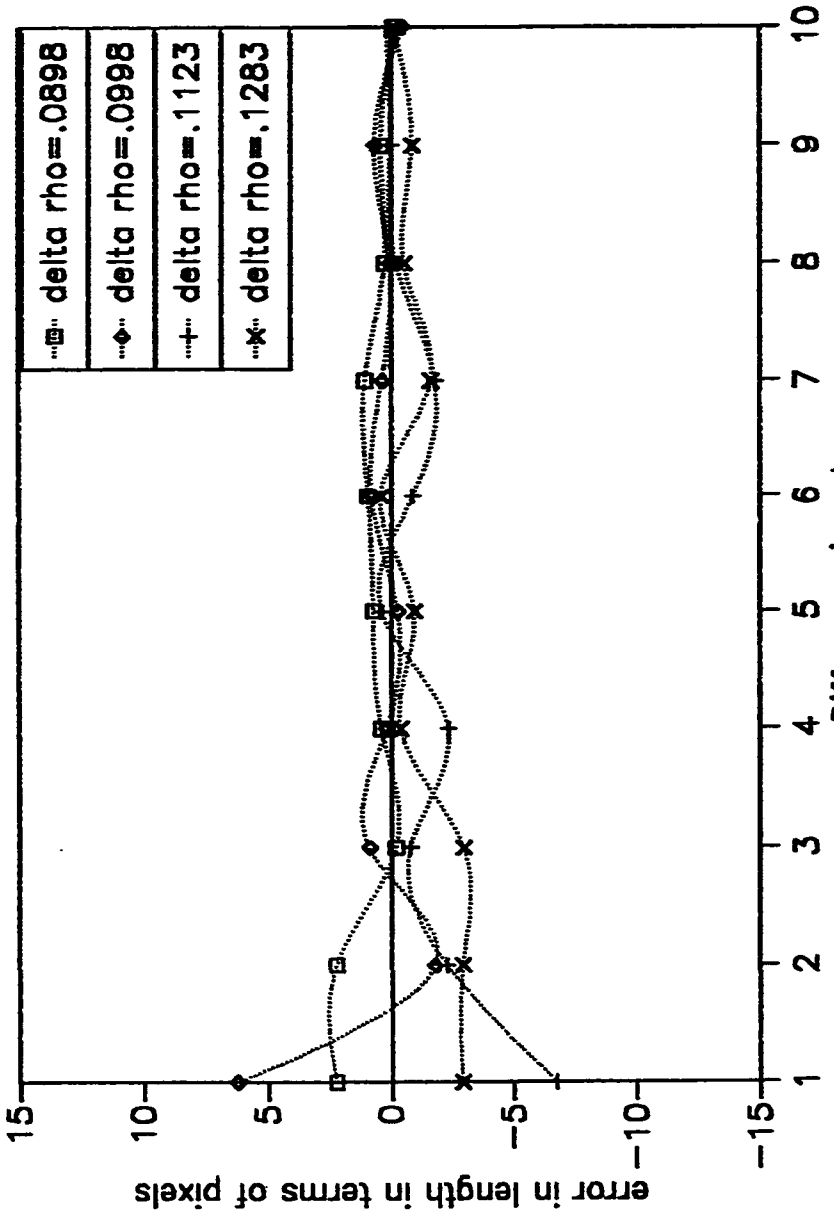


Fig 5.3.2.b Variation in error in $l_{computed}$ vs d_{θ_1}, for $\Delta\theta$ = 0.7087 and different $\Delta\rho$. N = 128. θ_{actual} = 25°, ρ_{actual} = 90.

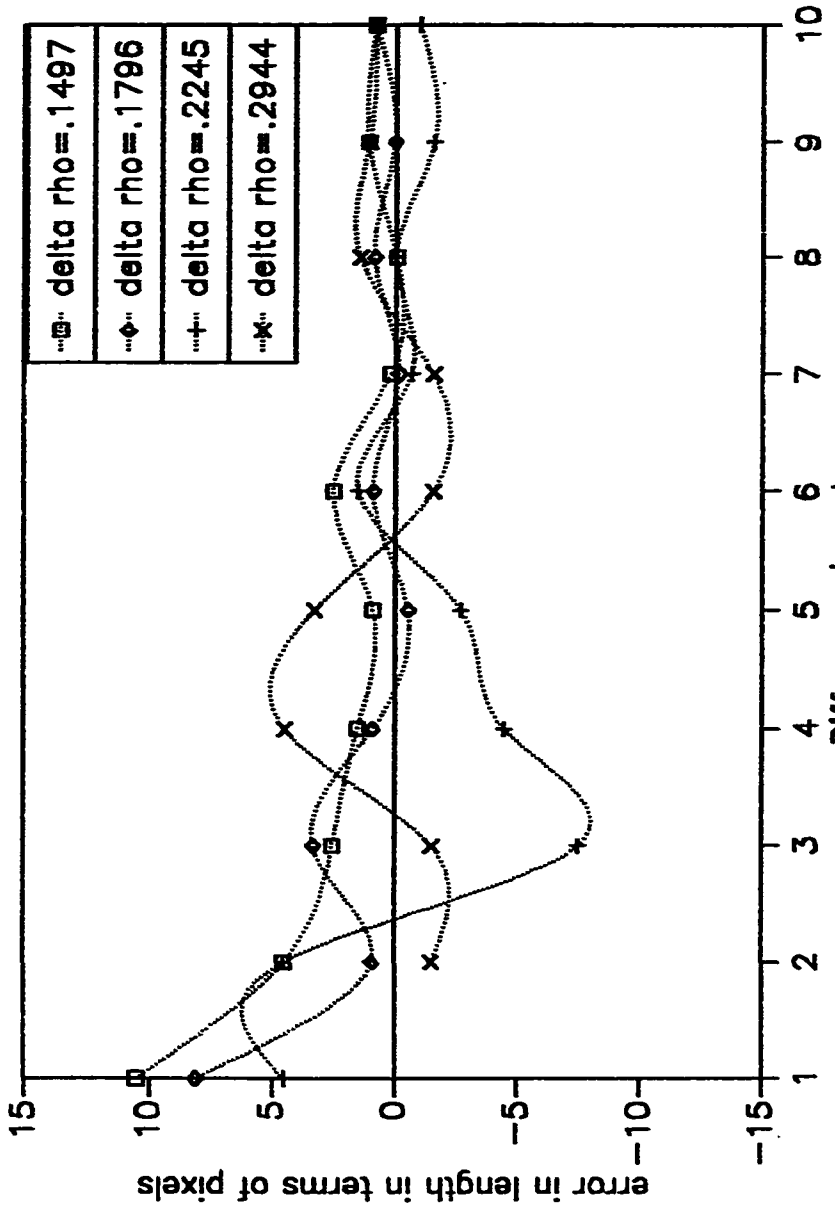


Fig 5.3.3.a Variation in error in $l_{computed}$ vs $d_{e,r}$ for $\Delta\theta$ = 0.7087 and different $\Delta\rho$. N = 128, θ_{actual} = 45°, ρ_{actual} = 90.

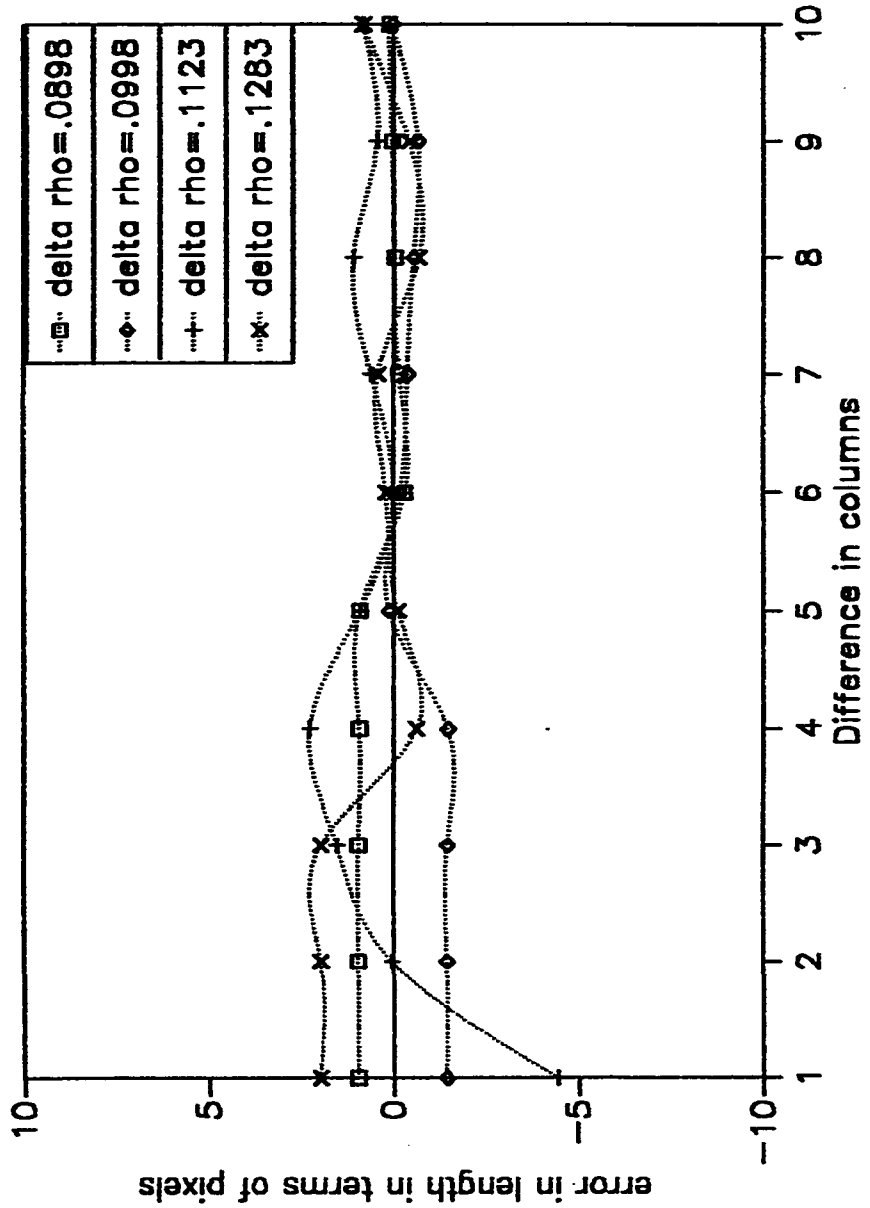


Fig 5.3.3.b Variation in error in $l_{computed}$ vs $d_{\theta,r}$ for $\Delta\theta = 0.7087$ and different $\Delta\rho$. $N = 128$, $\theta_{actual} = 4.5^{\circ}$, $\rho_{actual} = 90$.

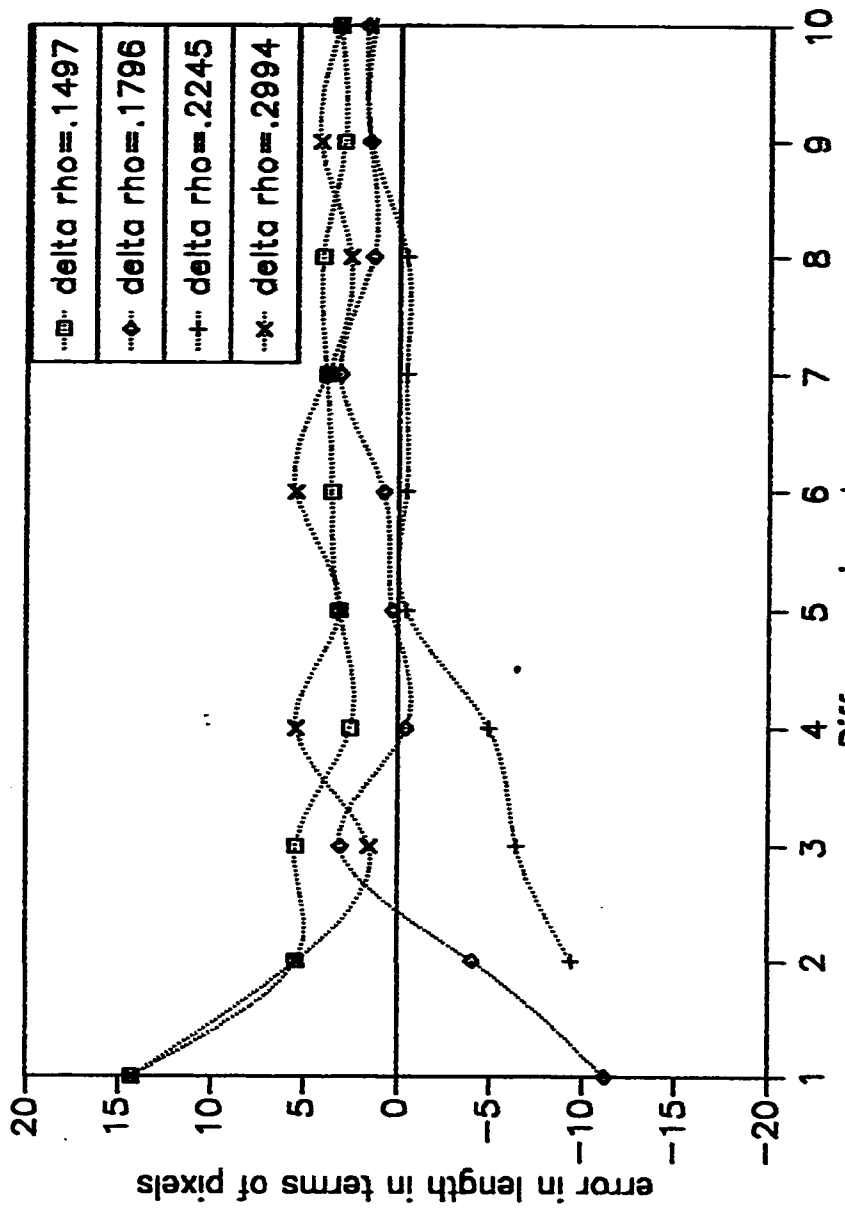


Fig 5.3.4.a Variation in error in $l_{computed}$ vs d_{θ_1}, for $\Delta\theta = 0.7087$ and different $\Delta\rho$. $N = 128$, $\theta_{actual} = 65°$, $\rho_{actual} = 90$.

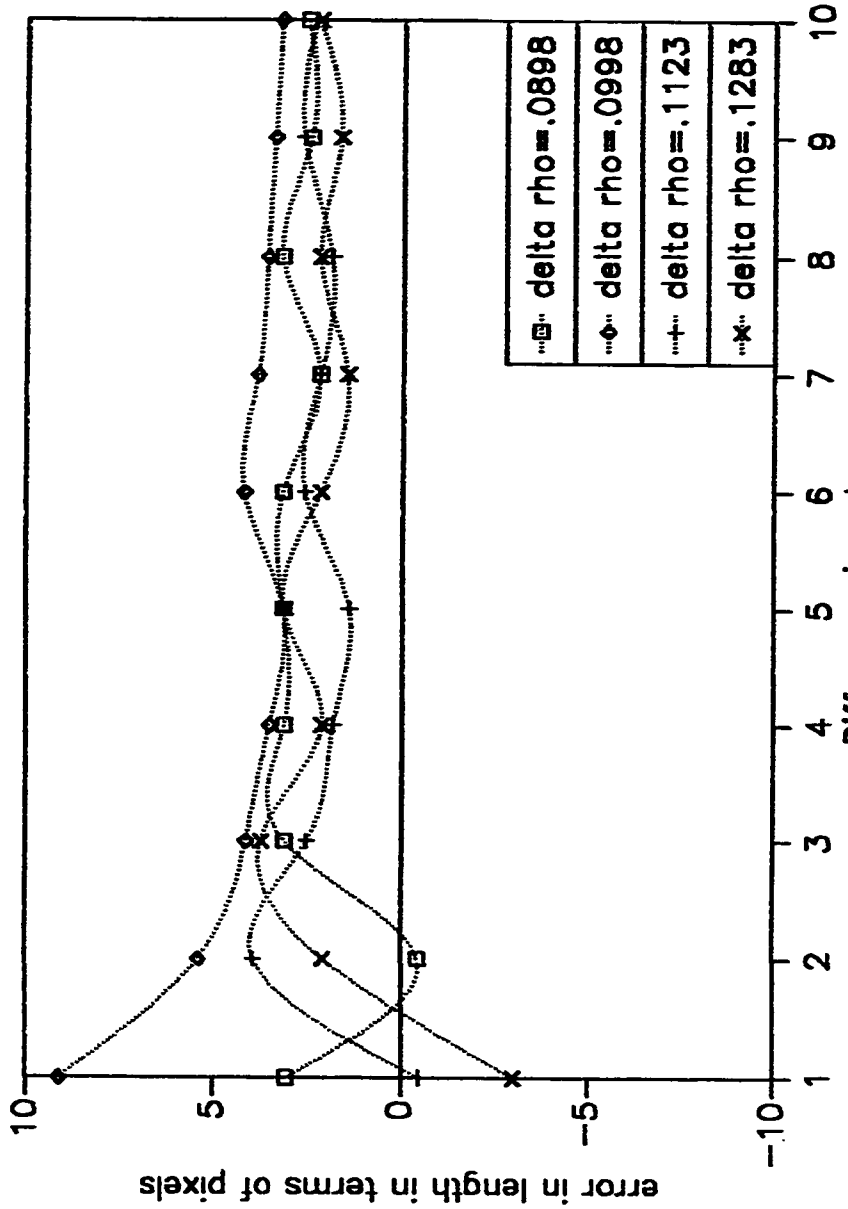


Fig 5.3.4.b Variation in error in $l_{computed}$ vs $d_{c,r}$ for $\Delta\theta = 0.7087$ and different $\Delta\rho$. $N = 128$, $\theta_{actual} = 65^{\circ}$, $\rho_{actual} = 90$.

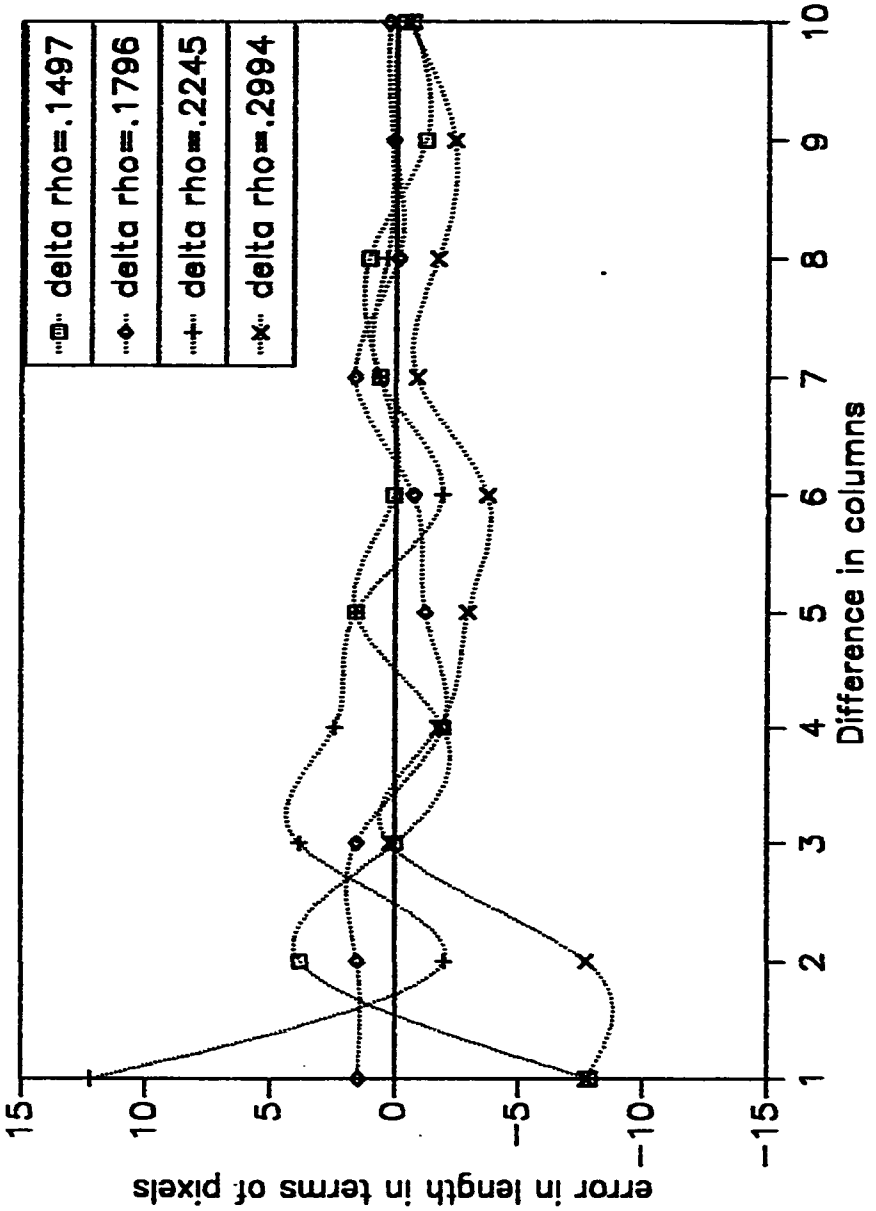


Fig 5.3.5.a Variation in error in $l_{computed}$ vs $d_{\theta,r}$ for $\Delta\theta = 0.7087$ and different $\Delta\rho$. $N = 128$, $\theta_{actual} = 75^{\circ}$, $\rho_{actual} = 90$.

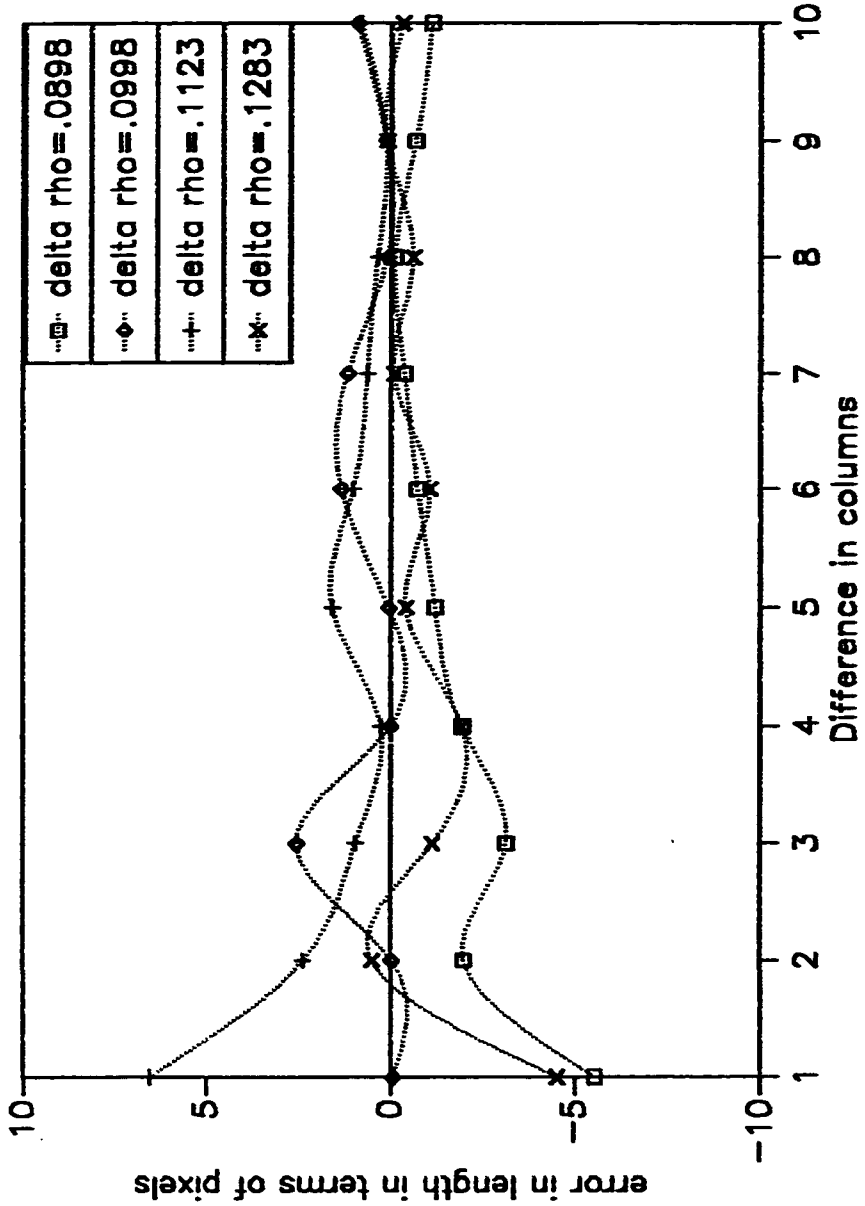


Fig 5.3.5.b Variation in error in $l_{computed}$ vs $d_{c,r}$ for $\Delta\theta = 0.7087$ and different $\Delta\rho$. $N = 128$. $\theta_{actual} = 75°$, $\rho_{actual} = 90$.

ordinates respectively. Using synthetic images, decreasing nature of error was obtained in the computed length with increasing $(\theta_r - \theta_q)$. It is apparent from fig 5.3.1 to fig 5.3.5 that when the columns are relatively far apart, an error of ± 3 to ± 5 pixels was obtained in the computed values of the length of the line segment.

An interesting study is the behavior of the errors averaged over a number of lines for different values of the parameters such as $d_{q,r}$, $\Delta\rho$ and $\Delta\theta$. The behavior of the error in the computed extremities and the length of the line segment (averaged over 71 lines), with respect to $d_{q,r}$, is shown in fig 5.3.6 to fig 5.3.11. Fig 5.3.6 to fig 5.3.8 are for $\Delta\rho = 0.9$ and varying values of $\Delta\theta$ whereas fig 5.3.9 to fig 5.3.11 are for $\Delta\rho = 0.2$ and varying values of $\Delta\theta$. The difference between the angles of the normals is in terms of the number of $\Delta\theta$'s apart. As is evident from these graphs, the error decreases as the difference between the columns used increases. It can also be concluded that as $\Delta\theta$ is made smaller, the errors increase while as $\Delta\rho$ is made smaller the errors decrease. This also verifies the predicted behavior of error given by eq 4.37 and eq 4.38.

Fig 5.3.12 and fig 5.3.13 illustrate the variation of error in the computed length (in terms of pixels, averaged over 71 lines) with respect to $\Delta\rho$ for different values of $\Delta\theta$ and for fixed $d_{q,r}$. These plots also corroborate the predicted behavior of the error in eq 4.37 and eq 4.38. As $\Delta\theta$ is made smaller, (which is usually done to increase the accuracy of $\theta_{predicted}$), the error in the computed extremities increases, thereby resulting in an increased error in the computed

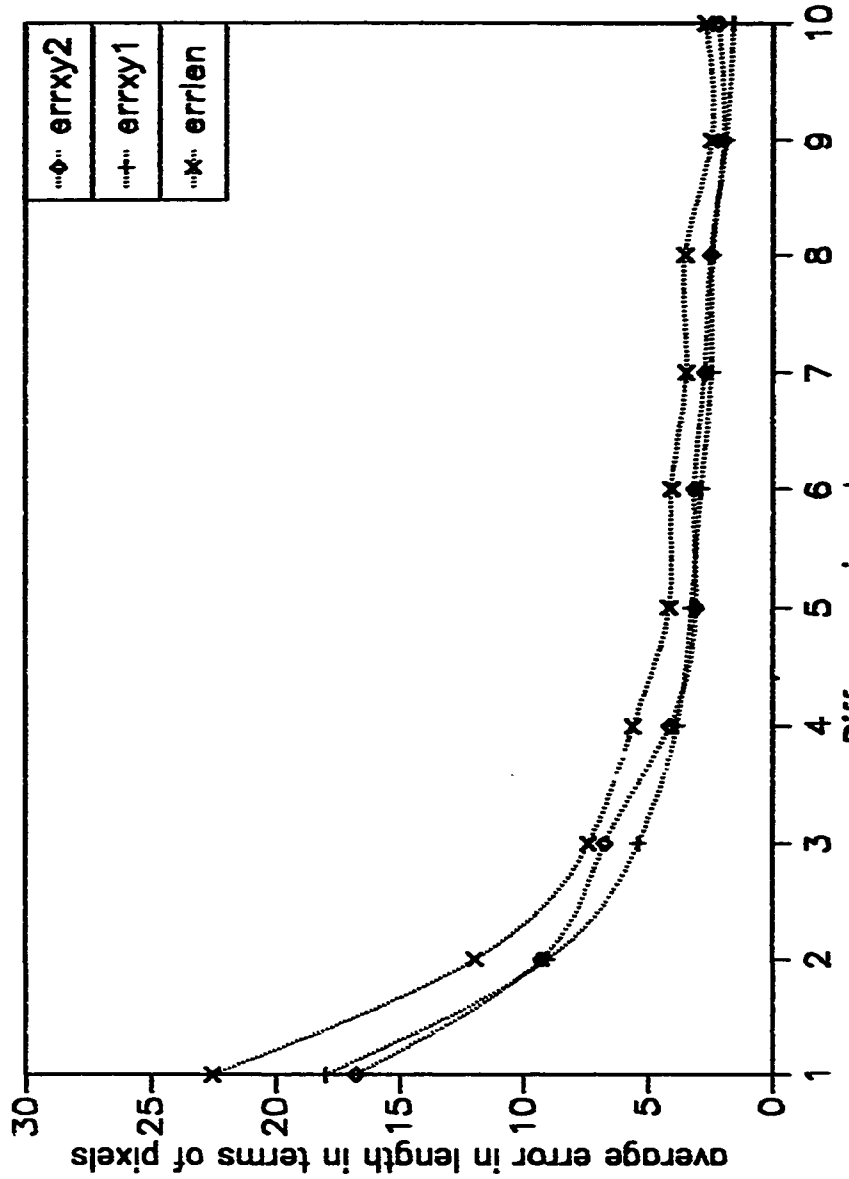


Fig 5.3.6 Variation in error in $l_{computed}$ and the computed co-ordinates of the endpoints, averaged over 71 lines vs $d_{c,r}$ for $\Delta\rho = 0.9$, and $\Delta\theta = 1.0$.

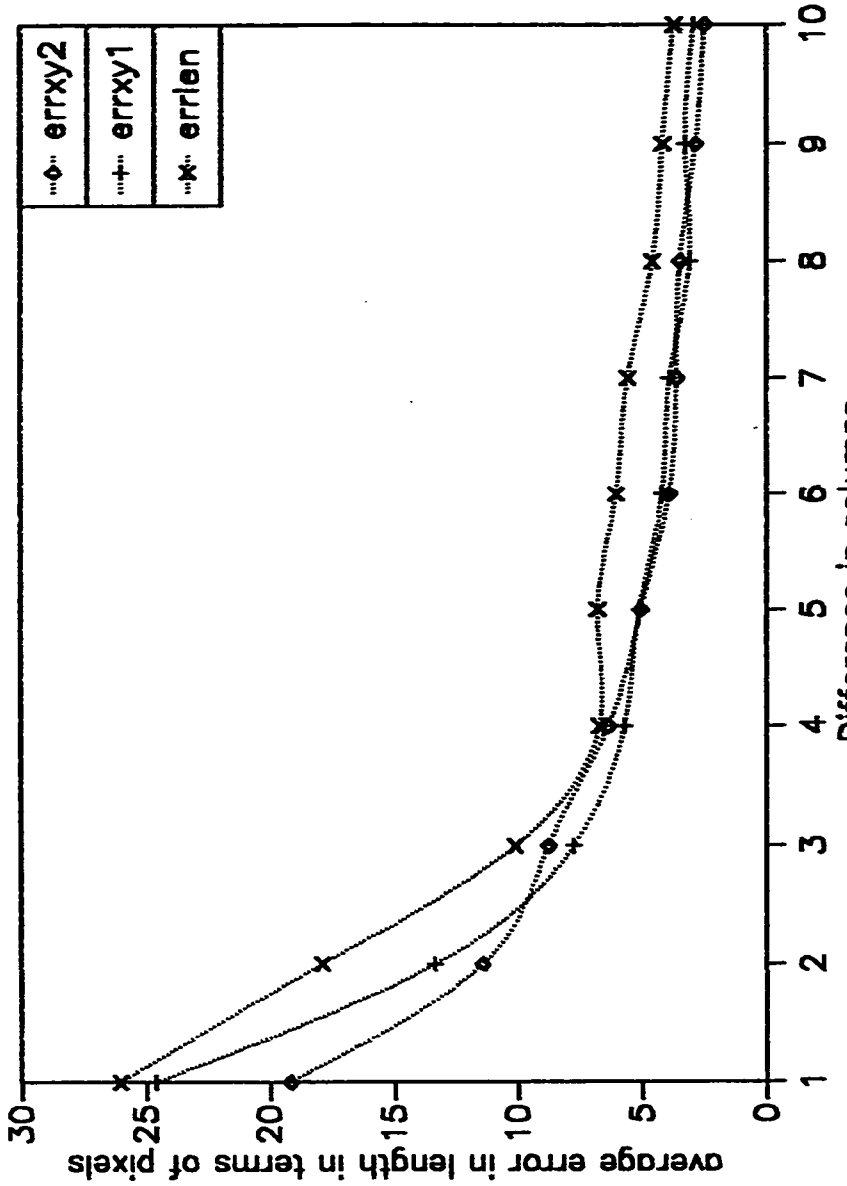


Fig 5.3.7 Variation in error in $l_{computed}$ and the computed co-ordinates of the endpoints, averaged over 71 lines vs $d_{x,y}$ for $\Delta\rho = 0.9$, and $\Delta\theta = 0.71$.

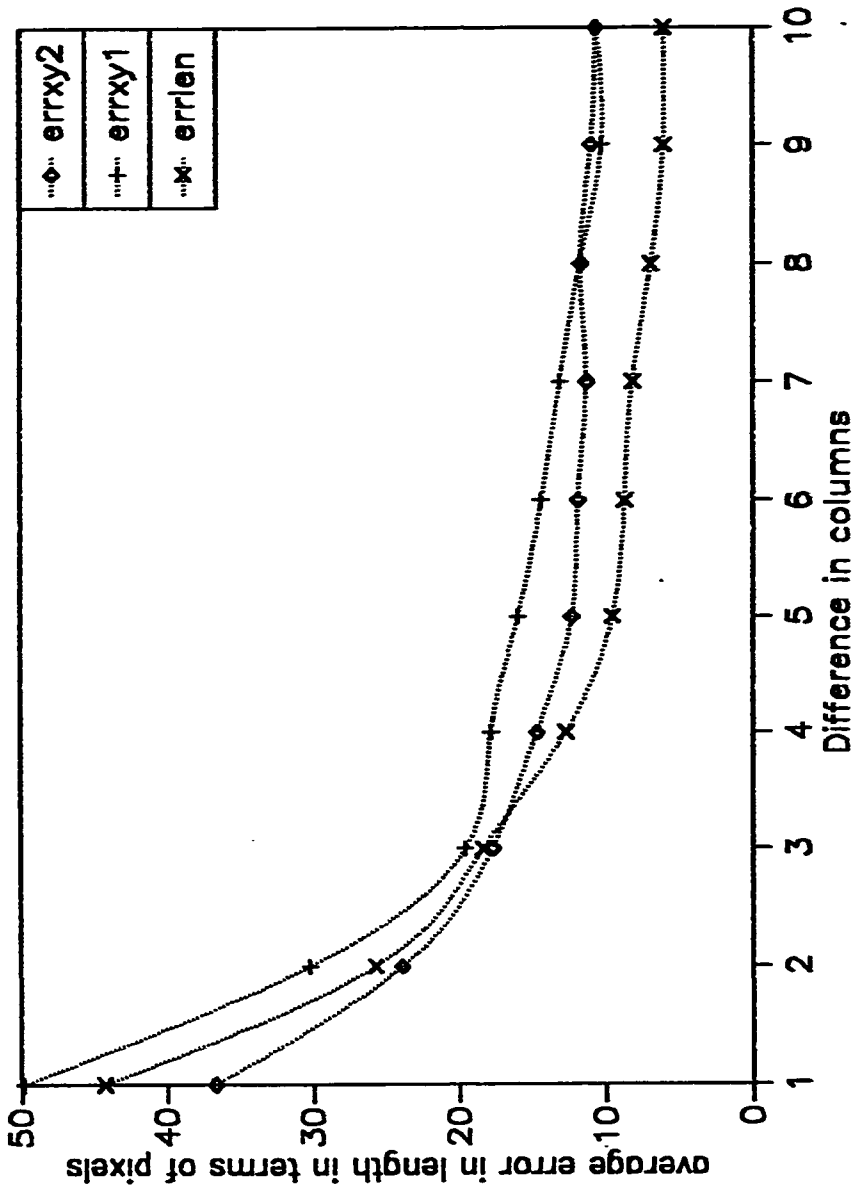


Fig 5.3.8 Variation in error in $l_{computed}$ and the computed co-ordinates of the endpoints, averaged over 71 lines vs $d_{r,r}$ for $\Delta\rho = 0.9$, and $\Delta\theta = 0.5$.

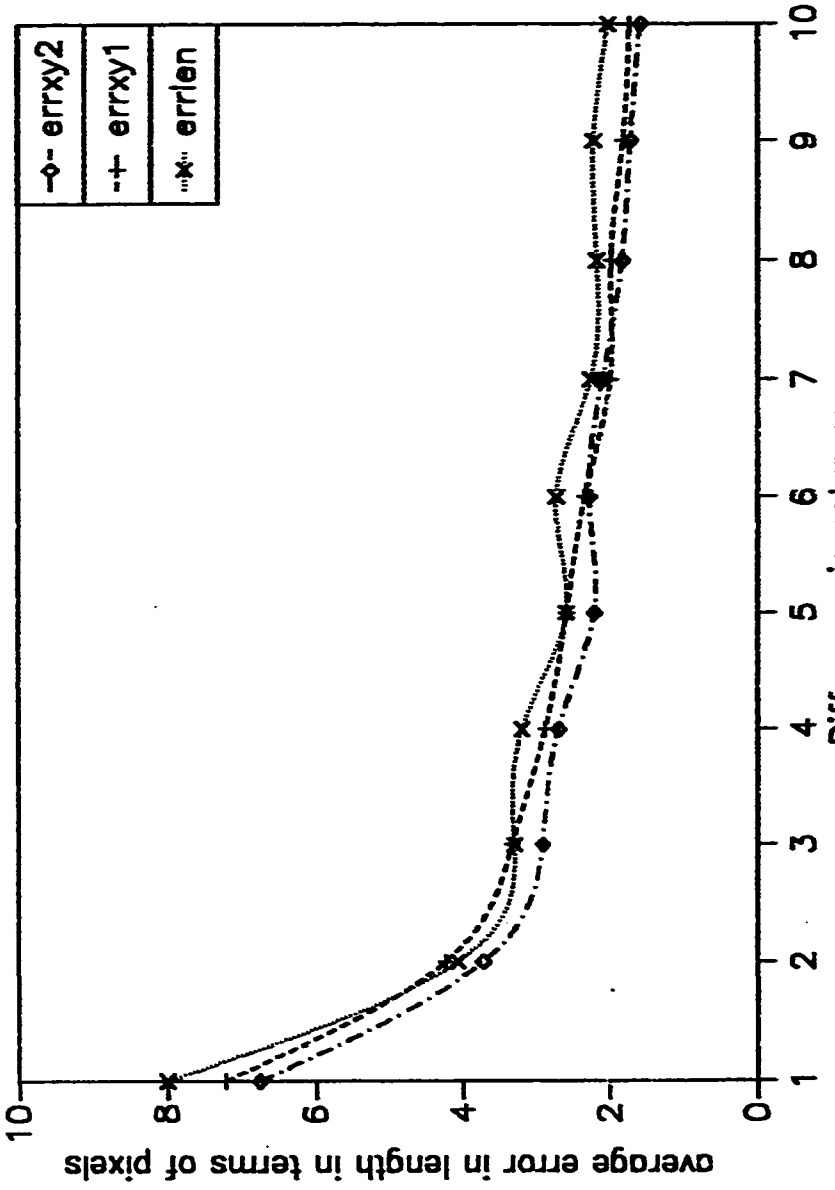


Fig 5.3.9 Variation in error in $l_{computed}$ and the computed co-ordinates of the endpoints, averaged over 71 lines vs $d_{i,i'}$ for $\Delta\rho = 0.2$, and $\Delta\theta = 0.71$.

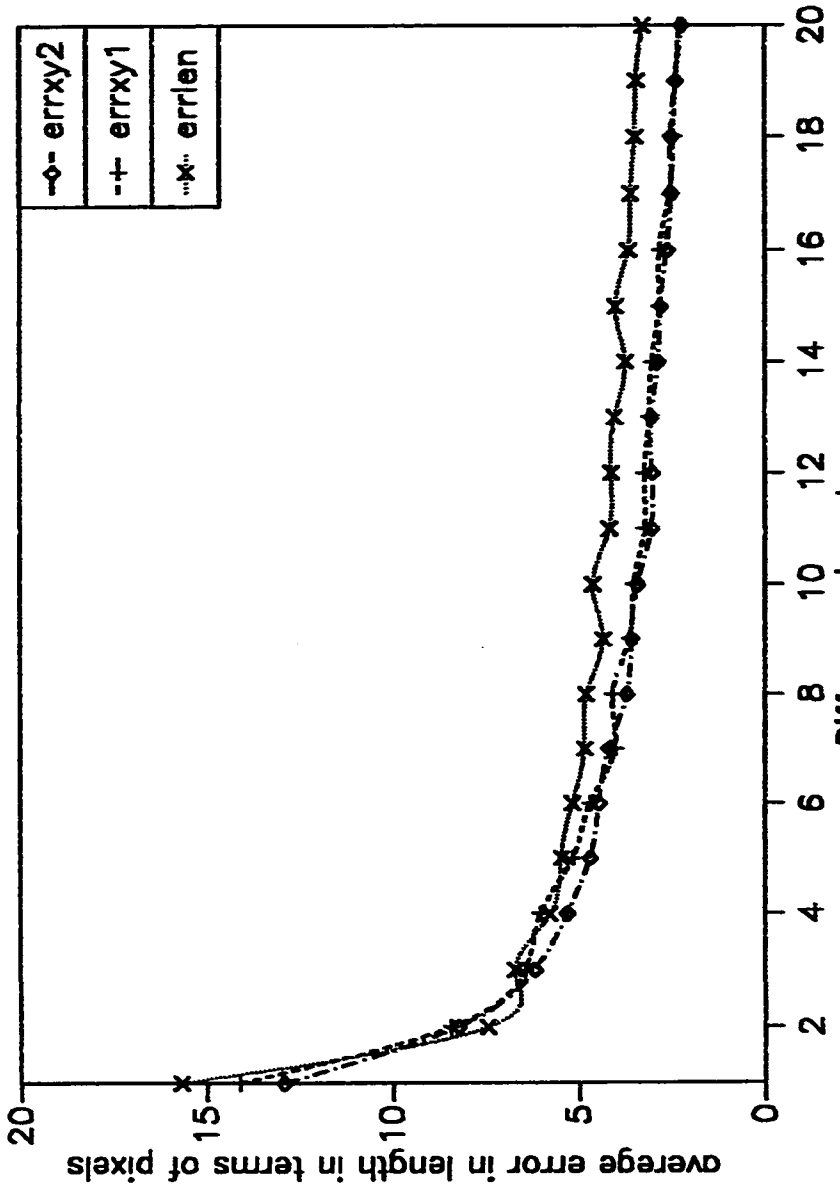


Fig 5.3.10 Variation in error in $l_{computed}$ and the computed co-ordinates of the endpoints, averaged over 71 lines vs $d_{x,y}$ for $\Delta\rho = 0.2$, and $\Delta\theta = 0.31$.

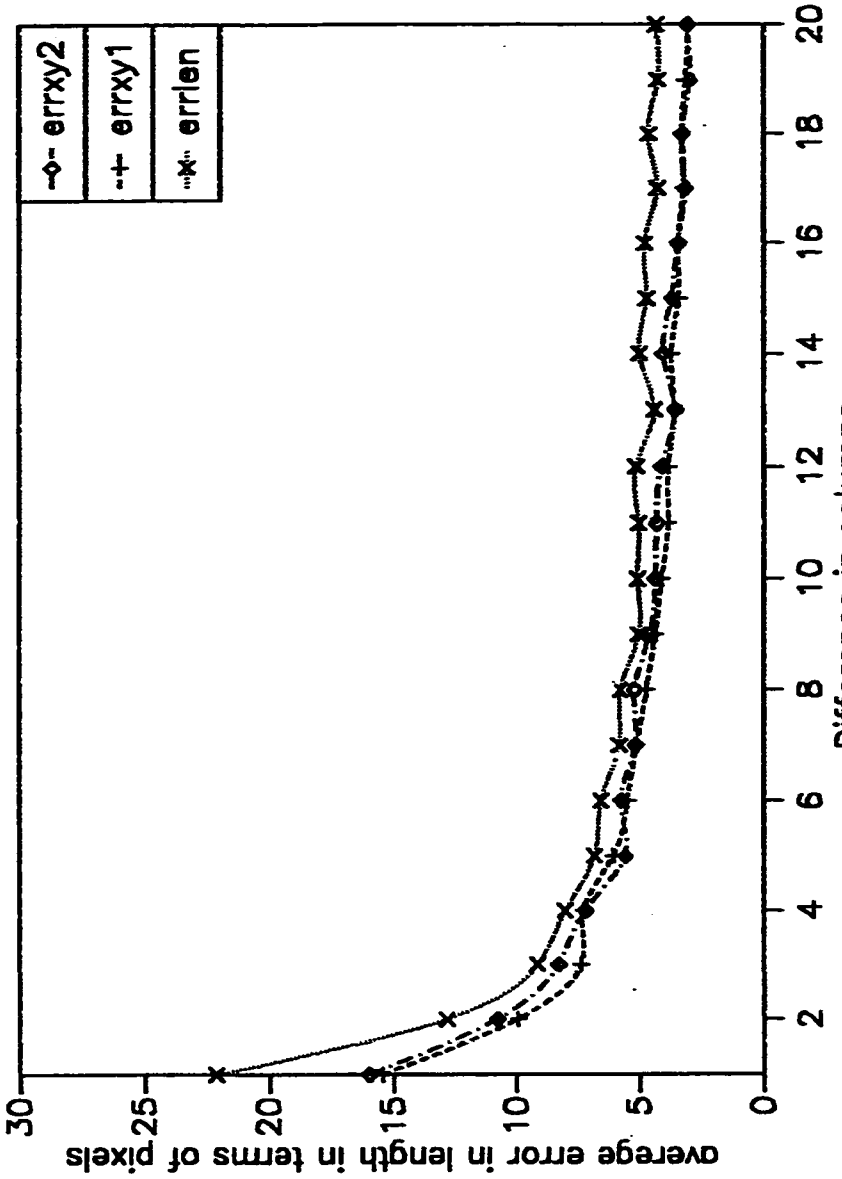


Fig 5.3.11 Variation in error in $l_{computed}$ and the computed co-ordinates of the endpoints, averaged over 71 lines vs $d_{r,r}$ for $\Delta\rho$ = 0.2, and $\Delta\theta$ = 0.23.

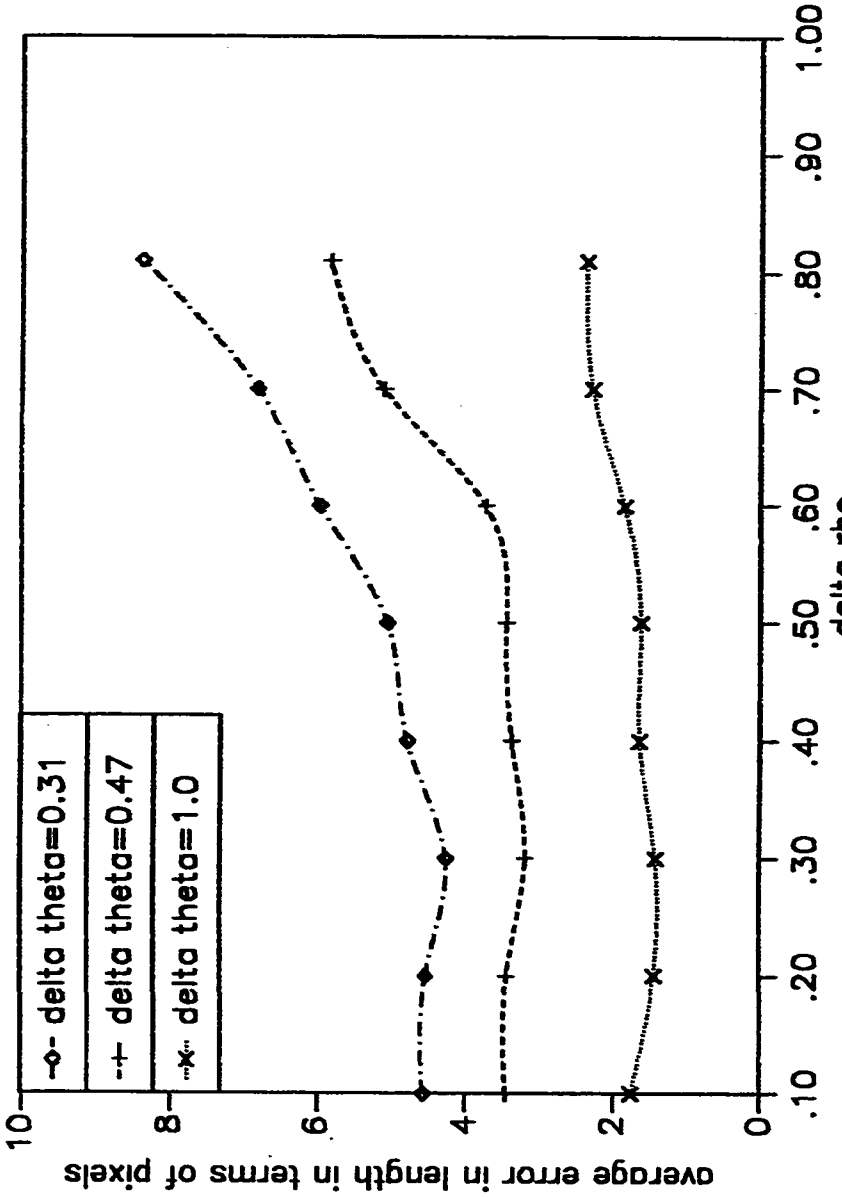


Fig 5.3.12 Variation in error in $l_{computed}$ averaged over 71 lines vs $\Delta\rho$ for fixed $d_{\theta,r}$ ($d_{\theta,r}$ = 10), and different $\Delta\theta$.

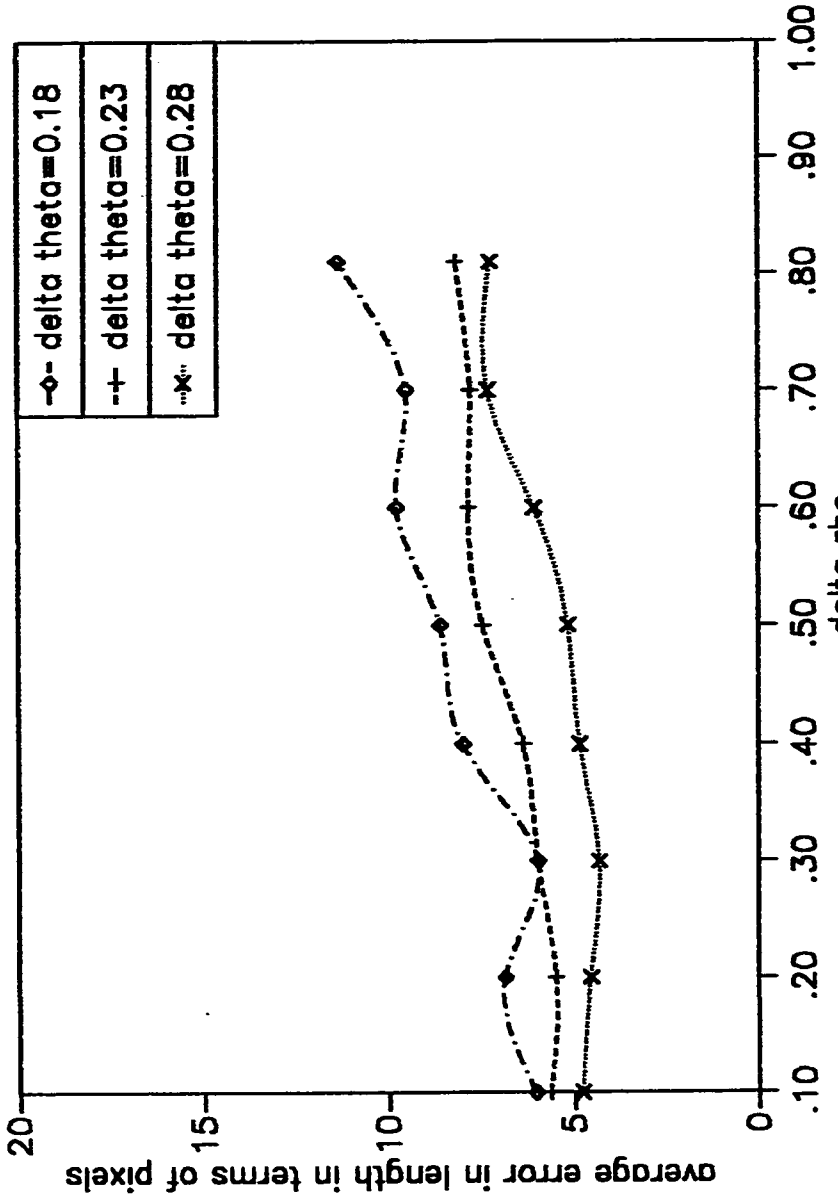


Fig 5.3.13 Variation in error in $l_{computed}$ averaged over 71 lines vs $\Delta\rho$ for fixed $d_{\theta,\rho}$, ($d_{\theta,\rho}$ = 10), and different $\Delta\theta$.

value of the length. Fig 5.3.14 shows this variation averaged over 141 lines. From eq 4.37 and eq 4.38, the errors increase as $\Delta\theta$ is made smaller because, with a decrease in $\Delta\theta$, the sine of the difference between the angles of the two columns decreases thereby leading to a increase in ε_x and ε_y.

Fig 5.3.15 and fig 5.3.16 depict the behavior in the computed value of the length (in terms of pixels, averaged over 71 lines), with respect to $\Delta\rho$ for various values of $d_{\varphi,r}$ and for a fixed value of $\Delta\theta = 0.71$. It is seen from the figures that the average error in length is higher for smaller values of $d_{\varphi,r}$. This further endorses the behavior of error predicted by eq 4.37 and eq 4.38. From fig 5.3.6 to fig 5.3.16, it is obvious that the extremities of a line segment can be detected accurately with an average error of ± 3 to ± 5 pixels for particular values of $\Delta\rho$ and $\Delta\theta$. These values of $\Delta\rho$ and $\Delta\theta$ could lead to the optimal accumulator array size as discussed in the next section.

5.4 Optimal Accumulator Array Size

The approach for finding the optimal accumulator array size is based on the objective that, if the extremities of a line segment are determined accurately, then the (ρ, θ) parameters of a line segment can be determined using eq 4.40 and eq 4.41 respectively. An opening can be made in the goal of finding the

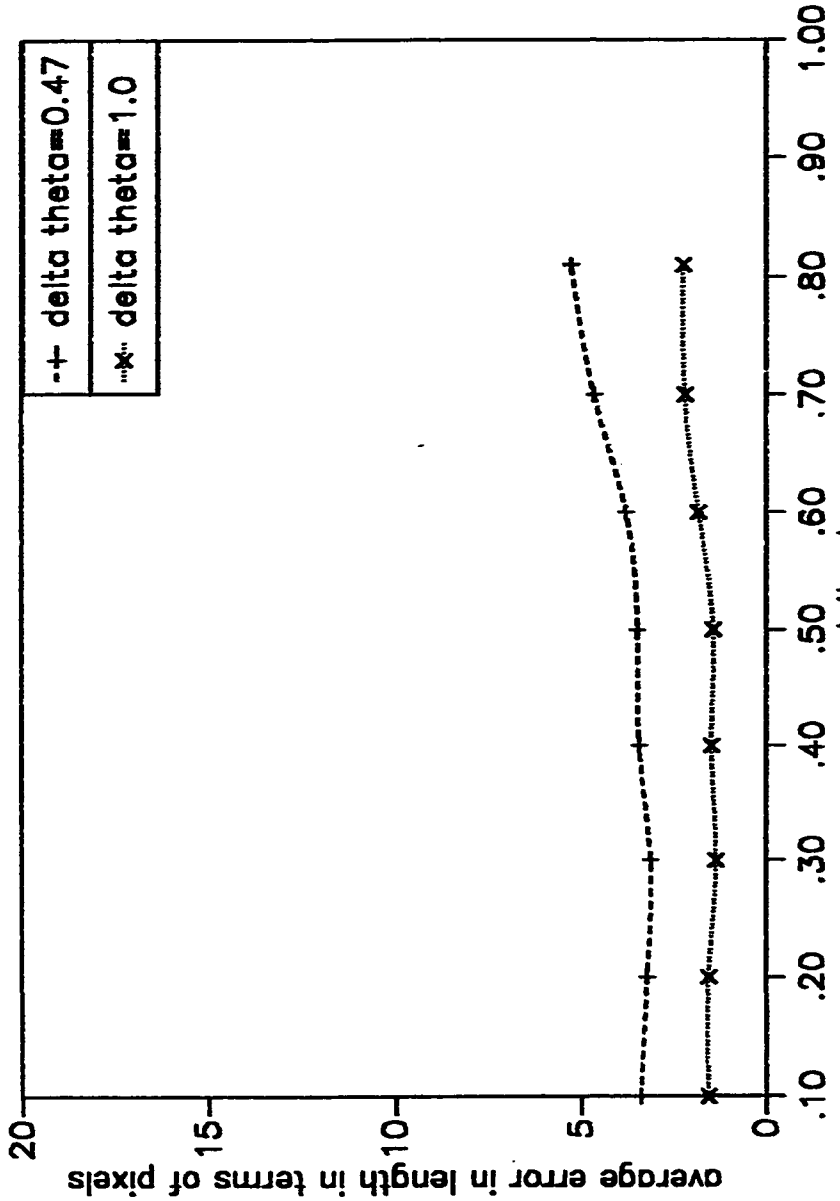


Fig 5.3.14 Variation in error in $l_{sampled}$ averaged over 141 lines vs $\Delta\rho$ for fixed $d_{\theta,r}$ ($d_{\theta,r}$ = 10), and different $\Delta\theta$.

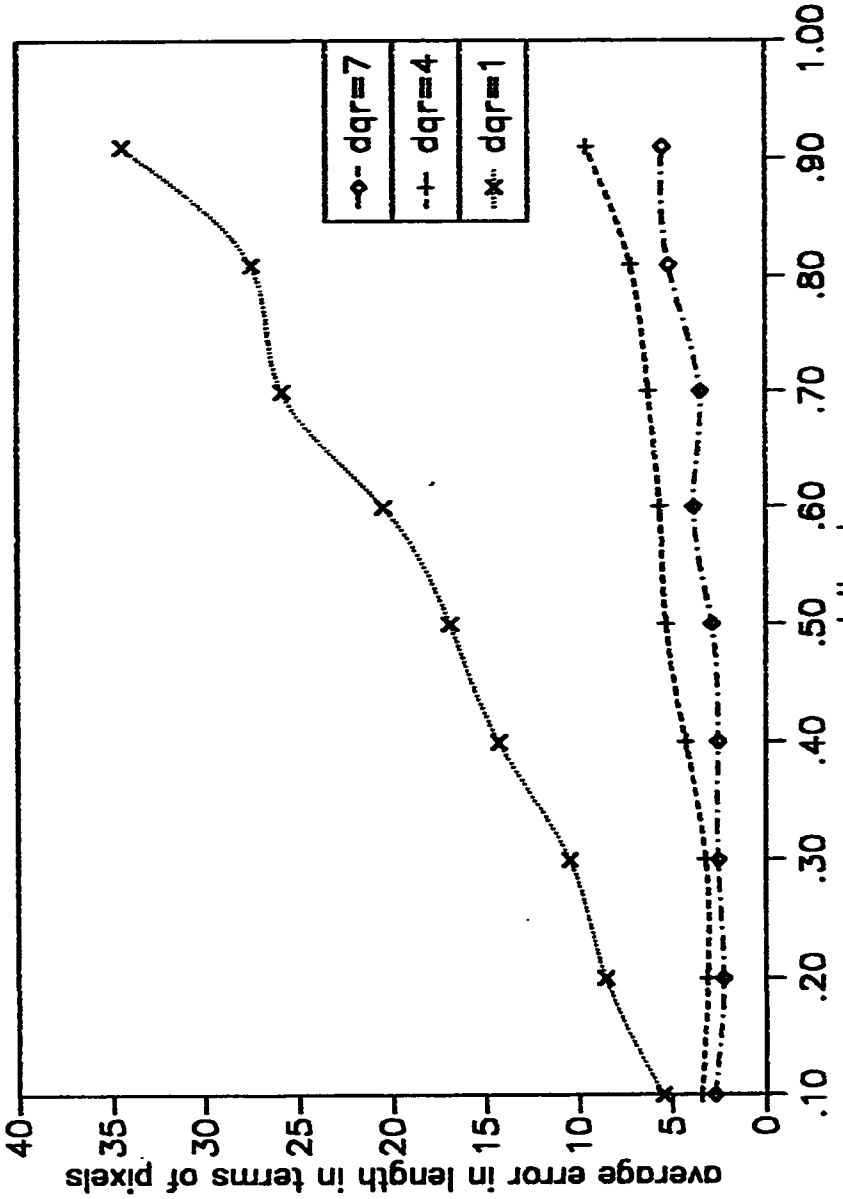


Fig 5.3.15 Variation in error in $l_{computed}$ averaged over 71 lines vs $\Delta\rho$ for fixed $\Delta\theta$, ($\Delta\theta$ = 0.71) and different $d_{q,r}$.

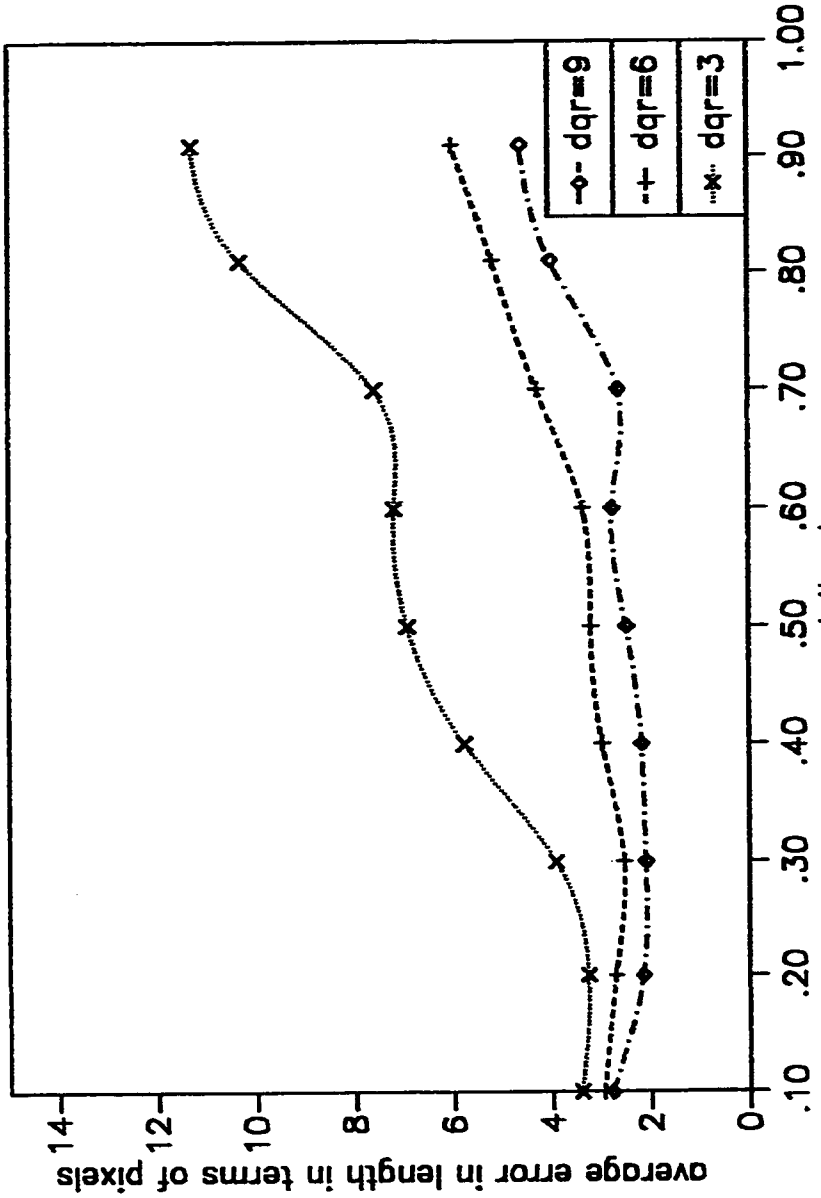


Fig 5.3.16 Variation in error in $l_{computed}$ averaged over 71 lines vs $\Delta\rho$ for fixed $\Delta\theta$, ($\Delta\theta$ = 0.71) and different $d_{q,r}$.

optimal accumulator array size (specifically for the resolution $\Delta\theta$), from the results discussed in the previous section. Comparing the results shown in fig 5.3.6, fig 5.3.12, fig 5.3.14 with all other figures which illustrate the similar results, it can be concluded that higher values of $\Delta\theta$ would give better results with respect to the error in the computed co-ordinates of the endpoints of the line segment. But using very high values of $\Delta\theta$ would lead to an increase in errors since, in this case, $\theta_{predicted}$ will not be determined accurately. Hence C_q and C_r will be selected incorrectly. In our simulations, a value of $\Delta\theta = 1$ was used (which is rather coarse) and a accuracy within ± 3 to ± 5 pixels in the computed co-ordinates of the endpoints is obtained.

One would argue that, a coarse value of $\Delta\theta$ might lead to an increased error in the value of predicted θ from the spread in the accumulator array. But this error in the predicted value of θ can be overcome, by computing the value of θ from the extremities of the line segment using eq 4.40. The computed extremities are more accurate for higher values of $\Delta\theta$. Another important advantage of using higher values of $\Delta\theta$ would be a reduction in the computational complexity of the HT. The computational complexity depends on the value of $\Delta\theta$. A high value of $\Delta\theta$ would mean fewer number of calculations of ρ using eq 2.4 for every feature point in the image.

With respect to the resolution $\Delta\rho$, it is seen from fig 5.3.12 to fig 5.3.16 that, the error is less for small values of $\Delta\rho$. This is also evident from eq 4.37 and eq 4.38, as the error in the computed x and y co-ordinates is directly proportional to $\Delta\rho$. But reducing $\Delta\rho$ to very small values could result in the phenomenon of

zero votes between non-zero votes and also bars containing cells of almost equal votes as discussed in sec 4.5.1. This can be overcome by using a value of $\Delta\rho$ given by eq 4.39; but this value of $\Delta\rho$ can be employed only after a initial estimate of the value of the θ of the line segment is known. Another way to overcome this could be the use of variable $\Delta\rho$ i.e., making $\Delta\rho$ a function of the sampled value of θ. Hence a trade-off has to be made between the accuracy desired and memory space available, since reducing $\Delta\rho$ increases the memory space requirements of the HT.

If the machine on which the HT is being computed can support large memory space requirements, then a small value of $\Delta\rho$ could be used, which reduces the error in the computed values of the extremities of the line segment. Once the extremities are computed precisely, then the normal parameters θ and ρ can be computed using eq 4.40 and eq 4.41 respectively.

5.5 Complete Line Segment Description in presence of noise

Noise plays a very critical role in determining the accuracy with which the extremities of a line segment can be computed using the HT. An approach to tackle the problem of noise in computing the extremities was described in sec 4.6. From the discussion in sec 4.6, it can be concluded that the accuracy of the computed extremities depends on several factors which interact among themselves in influencing the accuracy of the computed extremities. These

factors could be listed as the resolutions $\Delta\rho$ and $\Delta\theta$, the correction factor η^{k}_{correc}, the number of consecutive non-zero entries n_{consec} for which the columns C_q and C_r are scanned, the angle between the two columns used in computation i.e., $|r - q|\,\Delta\theta$ and above all the nature of noise present in the image. Results using the approach discussed in sec 4.6 on synthetic images containing noise are presented in the subsequent discussion (these are shown in fig 5.5.1 to fig 5.5.7).

It is clear from these results that the accuracy of the computed parameters depends heavily on the factors listed above. The effect of the various factors is discussed below.

5.5.1 Effect of $\Delta\rho$ and $\Delta\theta$

The values of $\Delta\rho$ and $\Delta\theta$ influence the accuracy of the computed extremities. As $\Delta\theta$ is made large and $\Delta\rho$ is made smaller the errors should decrease as predicted by eq 4.37 and eq 4.38. But in the presence of noise, decreasing $\Delta\rho$ leads to a increase in the errors as shown in fig 5.5.3.a and fig 5.5.3.b. This is because in presence of noise, smaller values of $\Delta\rho$ result in scattered non-zero entries in the columns of A. This can be avoided by increasing the value of $\Delta\rho$, so as to make the spread of non-zero votes more uniform. The relation between the behavior of errors with respect to $\Delta\rho$ as given by eq 4.37 and eq 4.38, does not hold good effectively in case of images containing noise. This is because of the scattering of non-zero votes in A. When

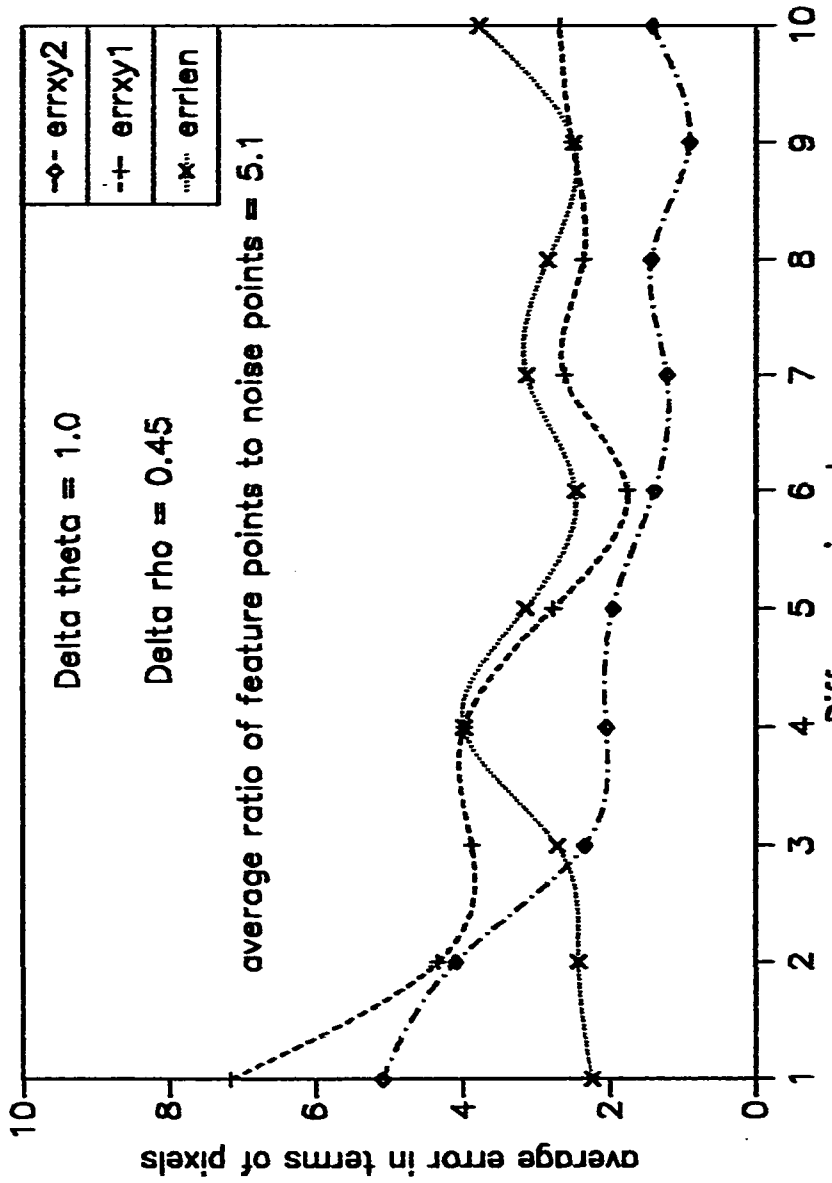


Fig. 5.5.1 Variation in error in $l_{computed}$ and computed extremeties averaged over 11 lines vs $d_{\theta,r}$ using $n_{center} = 3$ and $\eta^k_{error} = 2$

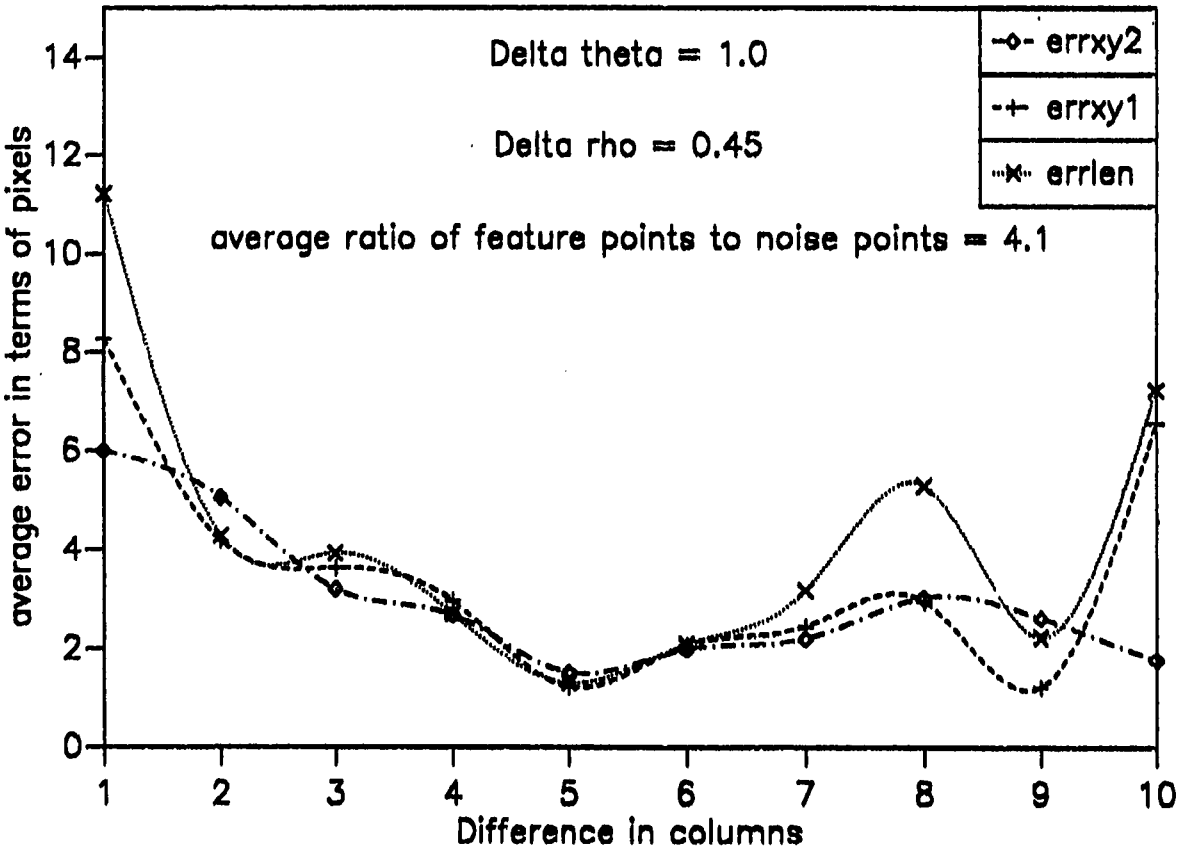


Fig 5.5.2 Variation in error in $l_{computed}$ and computed extremeties averaged over 11 lines vs $d_{q,r}$ using $n_{consec} = 3$ and $\eta^{k}_{corres} = 2$

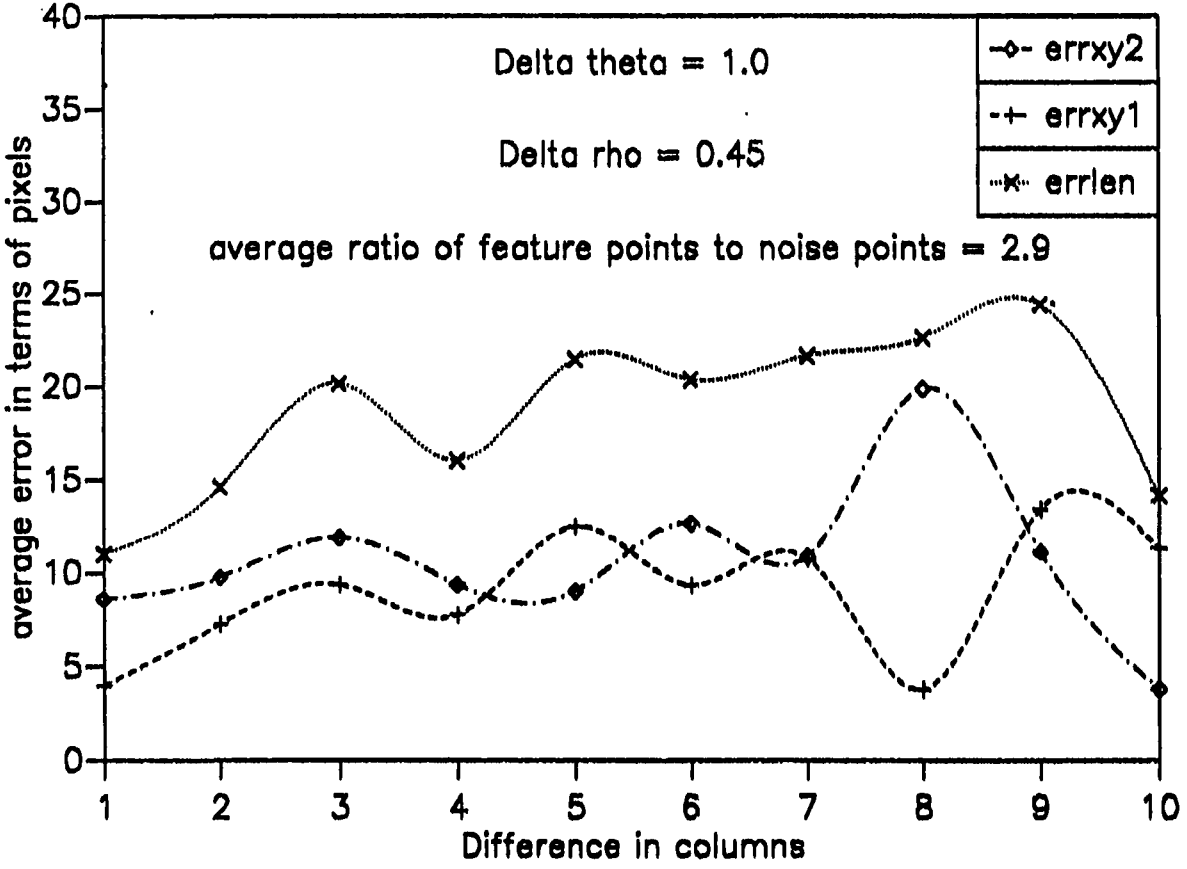


Fig 5.5.3.a Variation in error in $l_{computed}$ and computed extremeties averaged over 8 lines vs $d_{q,r}$ using $n_{consec} = 3$ and $\eta^{k}_{corres} = 2$

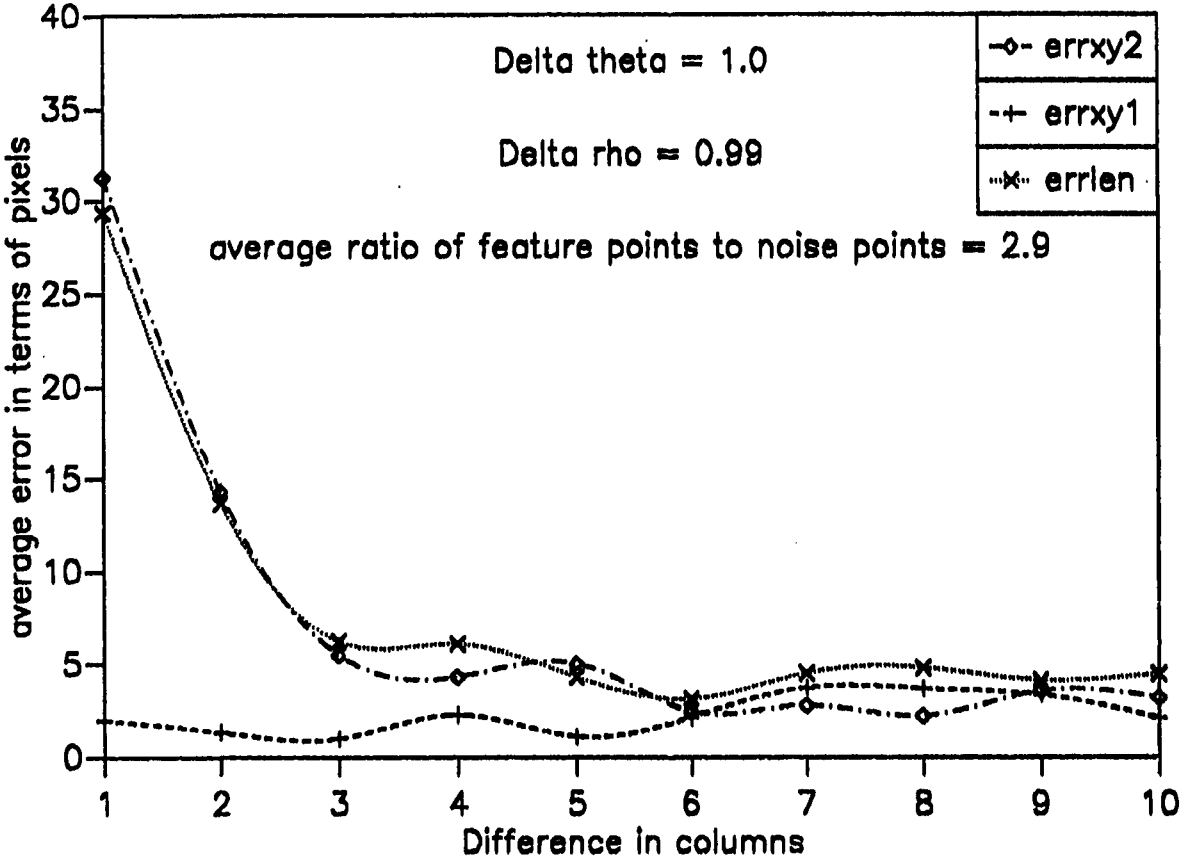


Fig 5.5.3.b Variation in error in $l_{computed}$ and computed extremeties averaged over 8 lines vs $d_{\theta,\rho}$ using $n_{consec} = 3$ and $\eta^{k}_{corres} = 2$

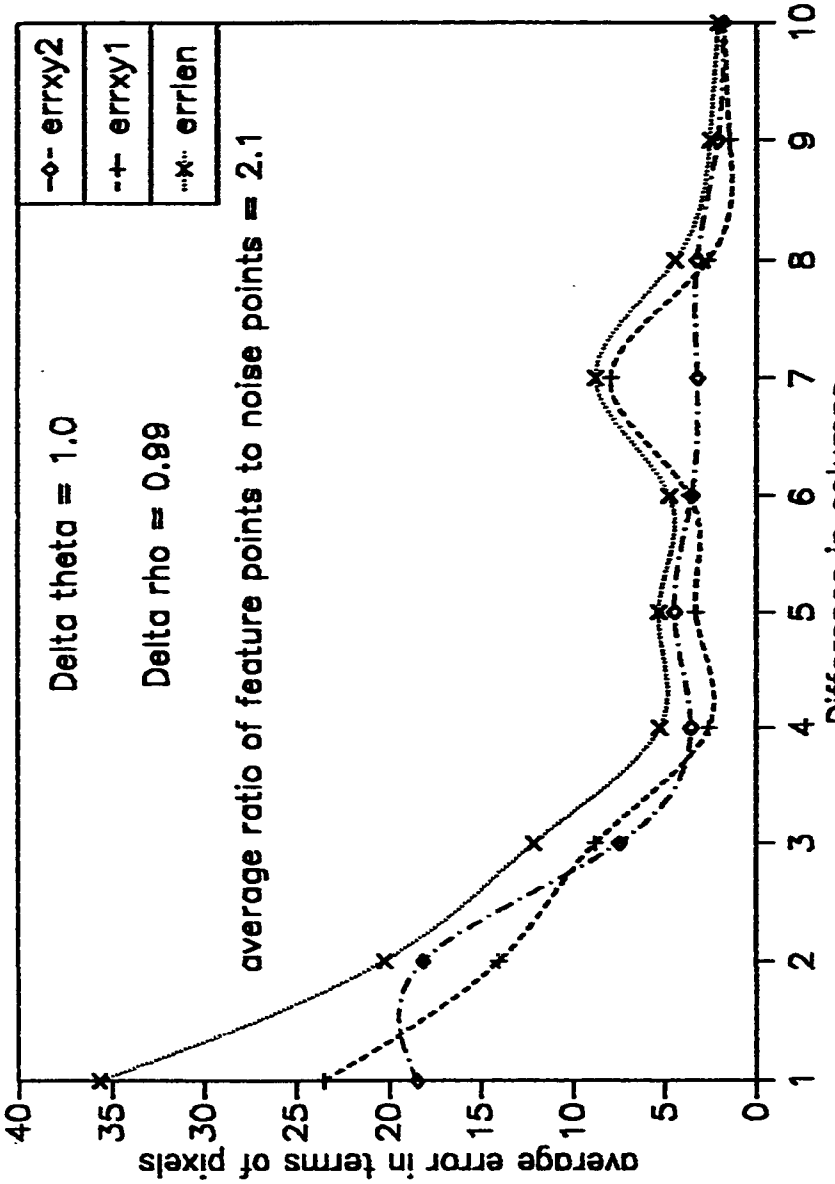


Fig 5.5.4.a Variation in error in $l_{computed}$ and computed extremities averaged over 16 lines vs $d_{c,r}$ using $n_{emiss} = 3$ and $\eta^{k}_{desired} = 2$

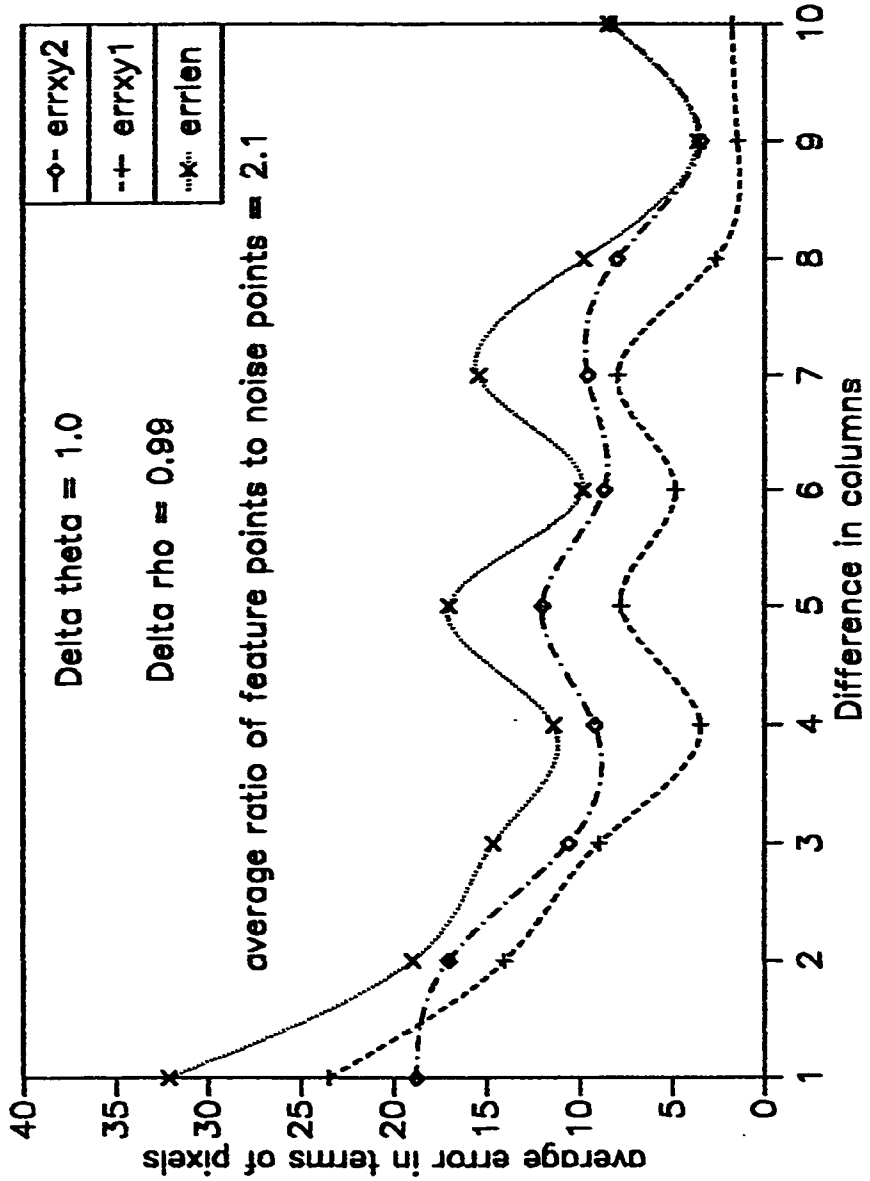


Fig 5.5.4.h Variation in error in $l_{computed}$ and computed extremation averaged over 16 lines vs d_{q_i} using $\eta_{consec} = 2$ and $\eta^{k}_{source} = 2$

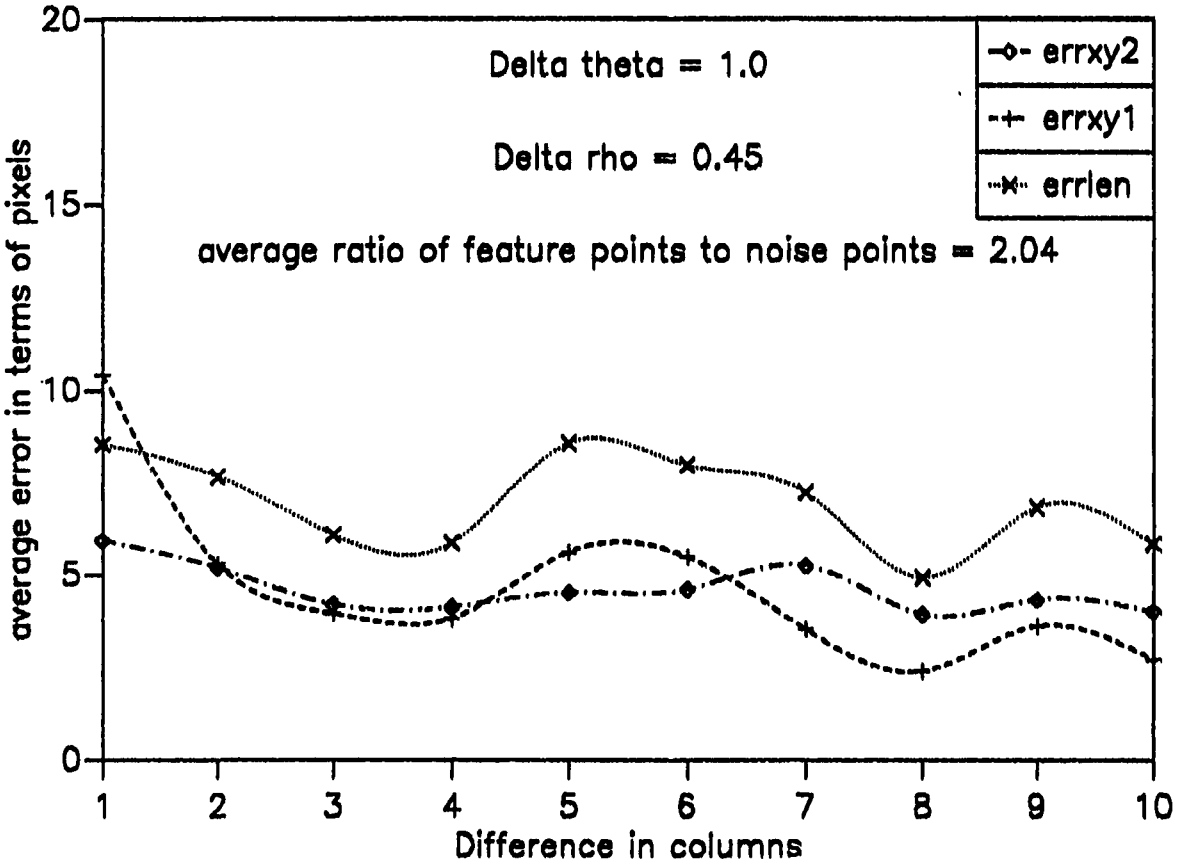


Fig 5.5.5.a Variation in error in $l_{computed}$ and computed extremeties averaged over 25 lines vs d_{q_1}, using $n_{consec} = 2$ and $\eta^k_{correc} = 2$

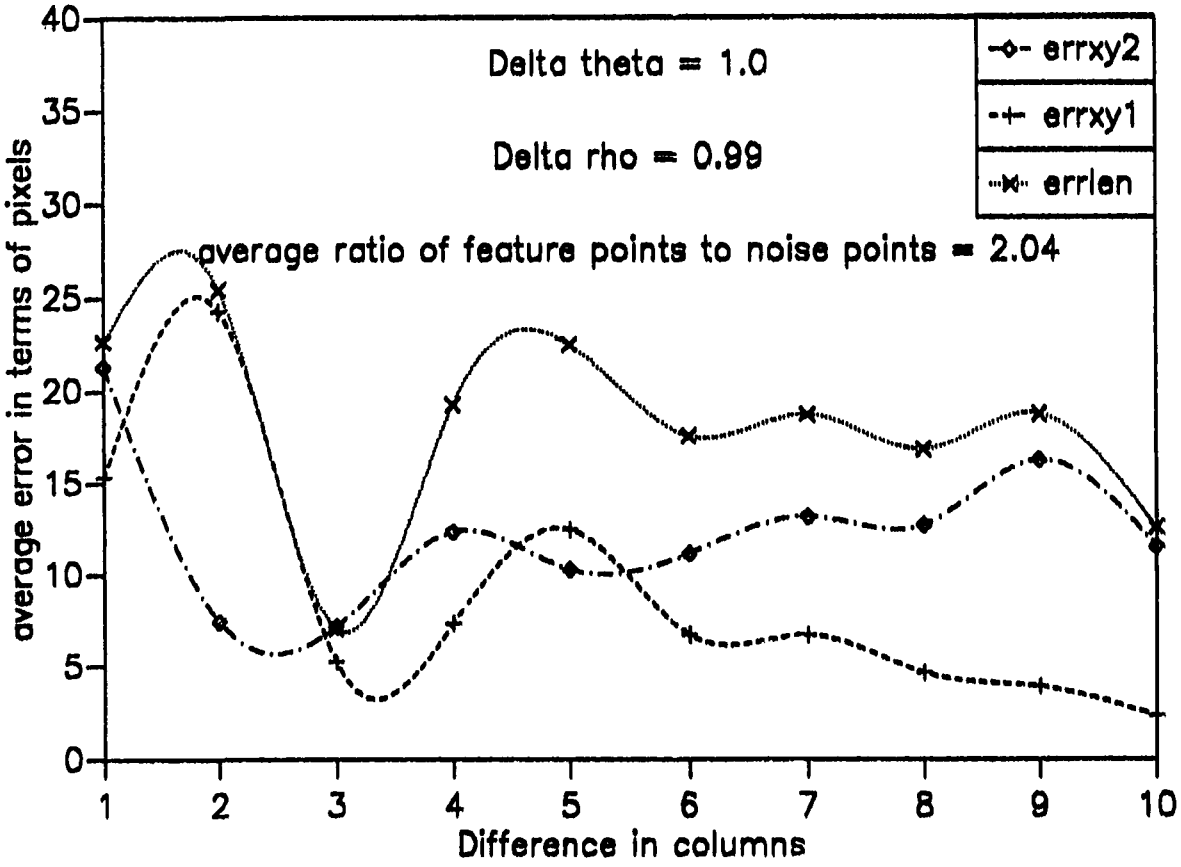


Fig 5.5.5.b Variation in error in $l_{computed}$ and computed extremeties averaged over 25 lines vs $d_{q,r}$ using $n_{consec} = 2$ and $\eta^{k}_{corres} = 2$

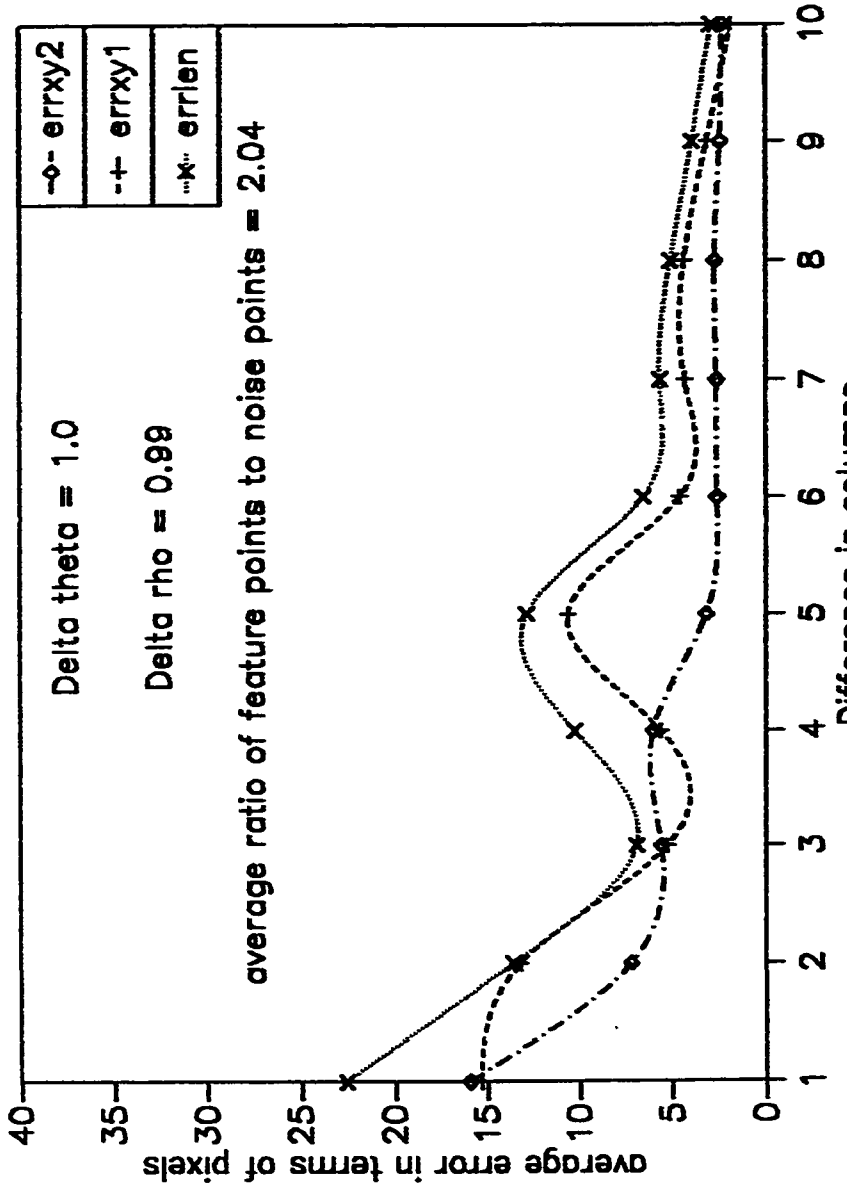


Fig 5.5.5.c Variation in error in $l_{computed}$ and computed extremities averaged over 25 lines vs $d_{x,y}$ using $n_{corner} = 4$ and $\eta^k_{corner} = 2$

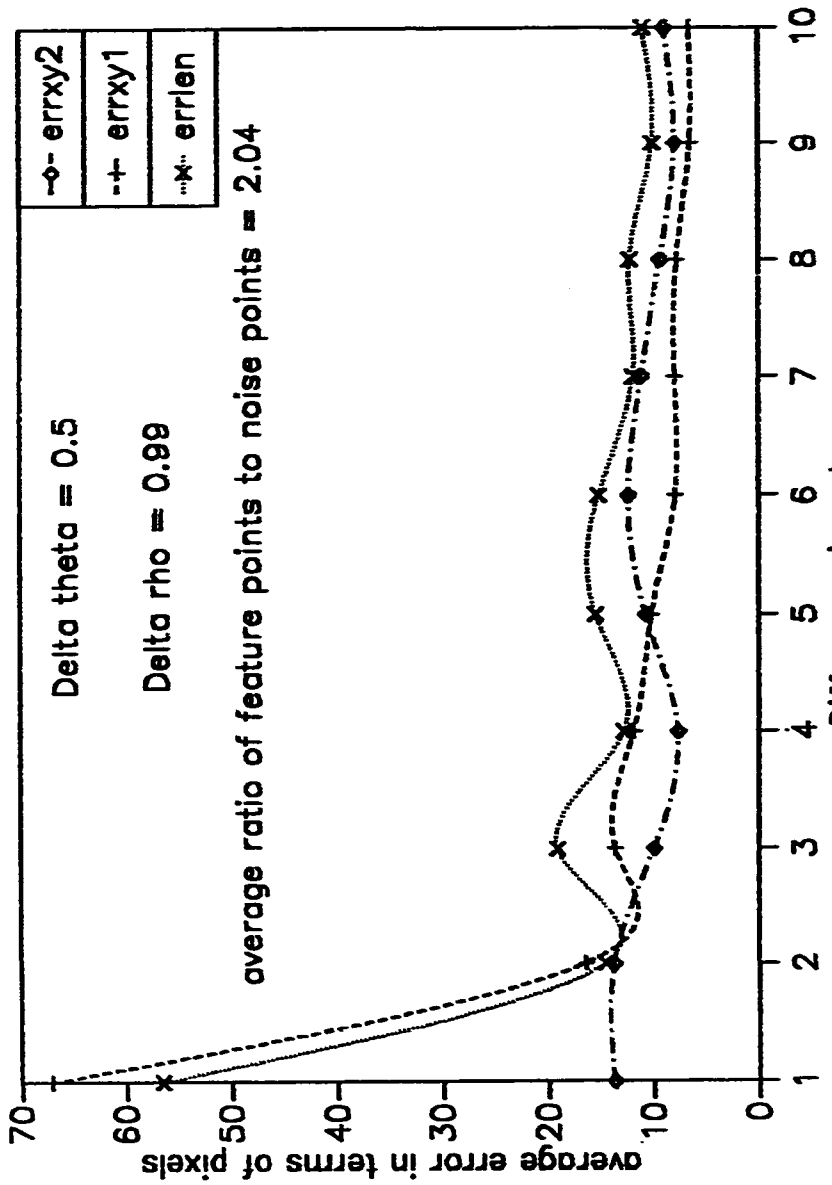


Fig 5.5.6 Variation in error in $l_{computed}$ and computed extremities averaged over 25 lines vs $d_{\theta,\rho}$ using $n_{consec} = 4$ and $\eta^{t}_{error} = 2$

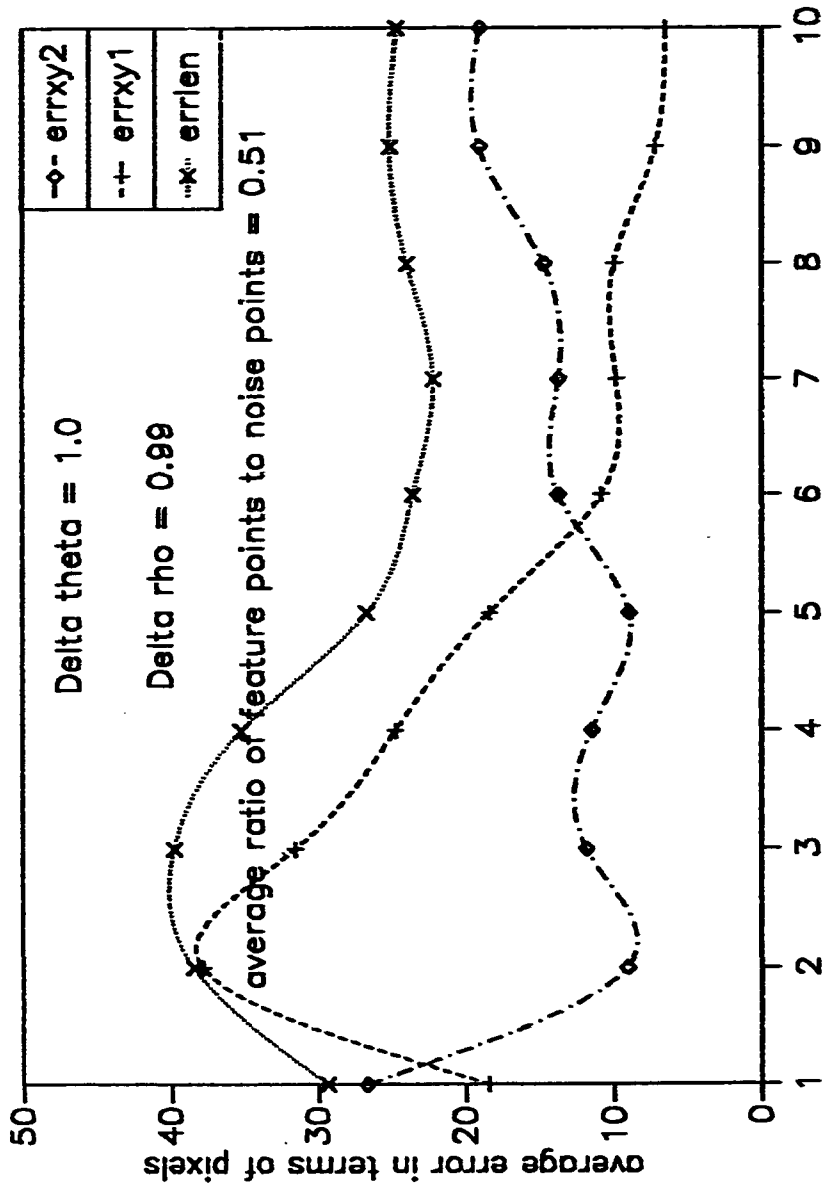


Fig 3.5.7.a Variation in error in $l_{computed}$ and computed extremities averaged over 25 lines vs $d_{x,y}$ using $n_{sensor} = 4$ and $\eta^1_{error} = 2$

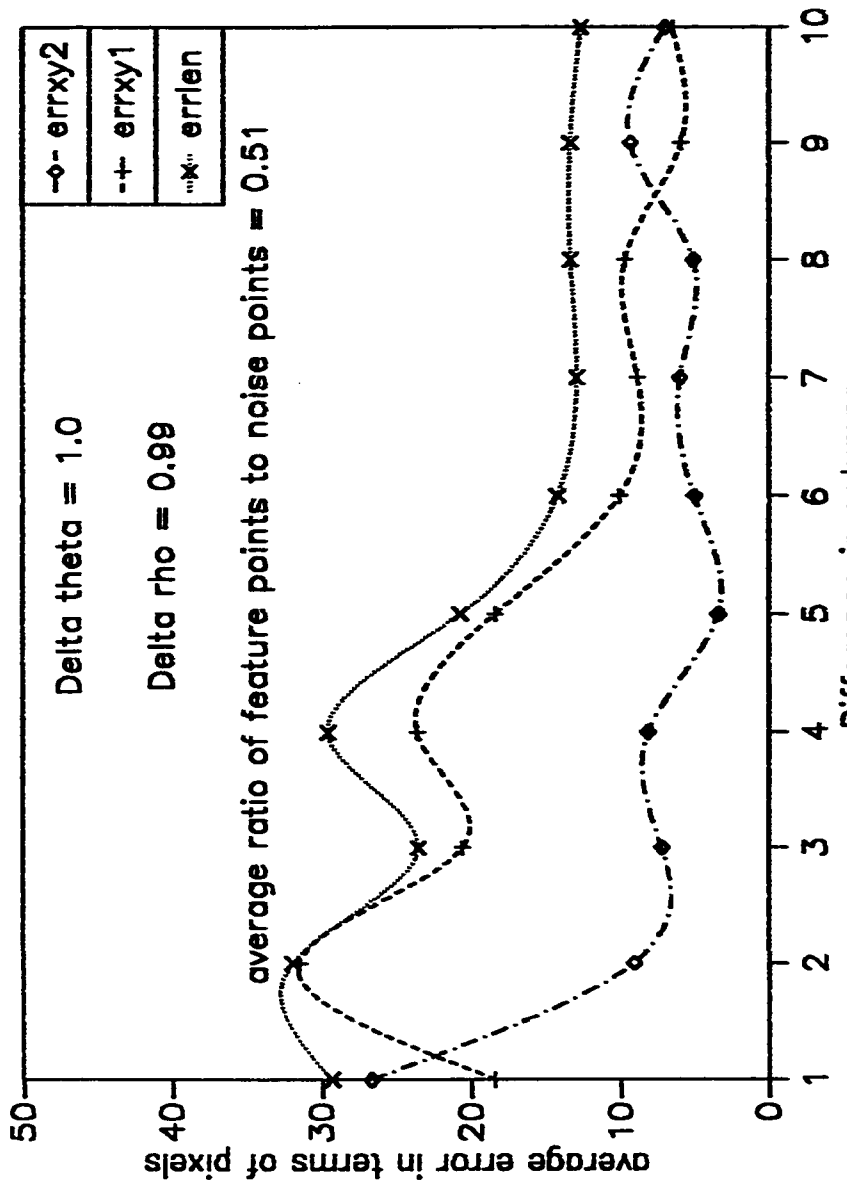


Fig 5.5.7.b Variation in error in $l_{computed}$ and computed extremeties averaged over 25 lines vs $d_{x,y}$ using $n_{senses} = 6$ and $\eta^{t}_{senses} = 2$

this type of scattering occurs, it is possible that the scanning for n_{consec} non-zero votes along any column may fail, which leads to high errors. The condition of n_{consec} non-zero votes has to be satisfied in order to ascertain that the votes are due to the feature points rather than the noise points. The effect of the value of n_{consec} is discussed in the next section. The predicted behavior of errors in eq 4.37 and eq 4.38 with respect to $\Delta\theta$ still holds as shown in fig 5.5.6.

5.5.2 Effect of n_{consec}

The value of n_{consec} also affects the accuracy of the extremities computed from the spread the spread in A. The value of n_{consec} has to be such that, the detection of cells corresponding to the contributions from the feature points is accurate. The value of n_{consec} depends on the nature of the image i.e., the amount of noise present. If the image contains large amount of noise, then a higher value n_{consec} would be required and vice versa, as can be seen from the results shown in fig 5.5.7.a and fig 5.5.7.b. The value of n_{consec} is not only related to the amount of noise present in the image, but also to the value of $\Delta\rho$. Smaller values of $\Delta\rho$ would require large value of n_{consec} while larger values of $\Delta\rho$ would require small value of n_{consec}. This ensures that the cells containing votes from feature points are detected properly. Fig 5.5.2 to fig 5.5.4 illustrate the points discussed in this section.

5.5.3 Effect of the amount of noise

The amount of noise in an image is perhaps the most important factor which determines the accuracy of the computed parameters. The lesser the amount of noise the better is the accuracy. The effect of the amount of noise on the accuracy of the computed parameters is illustrated in fig 5.5.1 to fig 5.5.7. As the noise increases i.e., the ratio of the feature points to the noise points decreases, the accuracy detoriates. An improvement in the accuracy obtainable also depends on other factors such as the values of $\Delta\rho$, $\Delta\theta$ and n_{consec}.

From the results shown in fig 5.5.7, it can be concluded that the accuracy of the procedure to obtain the complete line segment description from the spread in the accumulator array, in presence of noise, depends on a variety of factors. These factors are related among themselves in rather a complex way and their effect on the accuracy of the computed parameters depends on how they interact among themselves. An effort has been made to obtain the complete line segment description accurately in presence of noise. This could be further refined to obtain more accurate complete line segment description in the presence of noise.

5.6 Results using Real Images

The proposed approach for obtaining the complete line segment description from the spread in the accumulator array was tested on a real world binary image. The image is binary since the pixels can have only two gray levels- either 0 (background) or 1 (image). Since real world images inevitably contain noise, the modification suggested for tackling the problem of noise in sec 4.6 was adopted. The real image tested is shown in fig 5.6.1. It contains a line segment with noise and its size is 128 x 128. The detected pattern is shown in fig 5.6.2.a and fig 5.6.2.b. Fig 5.6.2.a illustrates the detected line segment with $d_{q,r} \doteq 9$, where C_q and C_r are the columns used in the computation of endpoints with $q = \theta_{peak} \pm 5$. The resolutions used were $\Delta\theta = 1.0$ and $\Delta\rho = 0.99$ with $n_{consec} = 2$ and $\eta^k_{correc} = 2$. Fig 5.6.2.b shows the detected line segment with $d_{q,r} = 10$ and all other factors remaining the same as in fig 5.6.2.a.

From these results, it can be concluded that by employing the proposed approach, single line segments can be detected accurately by using large difference between C_q and C_r i.e., large $d_{q,r}$ and by choosing the values of n_{consec} and η^k_{correc} according to the values of $\Delta\rho$. Fig 5.6.3 illustrates the variation of error in the computed length and the extremities with the difference between columns i.e., $d_{q,r}$ for $\Delta\rho = 0.99$, $\Delta\theta = 1.0$, $n_{consec} = 2$ and $\eta^k_{correc} = 2$. As $d_{q,r}$ increases, the errors decrease as was the case in synthetic images. The accuracy of the proposed method can also be judged from the following comparision

(based on the results for the real image shown in fig 5.6.1) :

(i) ρ and θ determined from the peak in the accumulator array are $\rho_{predicted} = 59.369$ and $\theta_{predicted} = 47^{\circ}$ respectively.

(ii) ρ and θ computed from the extremities which are obtained from the spread in the accumulator array with $d_{q,r} = 9$ are $\rho_{computed} = 59.290$ and $\theta_{computed} = 46.969^{\circ}$ respectively.

(iii) ρ and θ computed from the extremities which are obtained from the spread in the accumulator array with $d_{q,r} = 10$ are $\rho_{computed} = 59.197$ and $\theta_{computed} = 46.961^{\circ}$ respectively.

From (i), (ii) and (iii) it is obvious that using the proposed method is advantageous than the conventional HT because, by using the conventional HT only ρ and θ of a line can be obtained, but by using the proposed approach, not only ρ and θ are computed accurately, but the precise location of the line segment in the image is also obtained from the computed extremities. In addition to this advantage, the computational complexity with respesct to construction of the accumulator array is reduced compared to the conventional HT, since coarser values of $\Delta\rho$ and $\Delta\theta$ are required in the new approach.

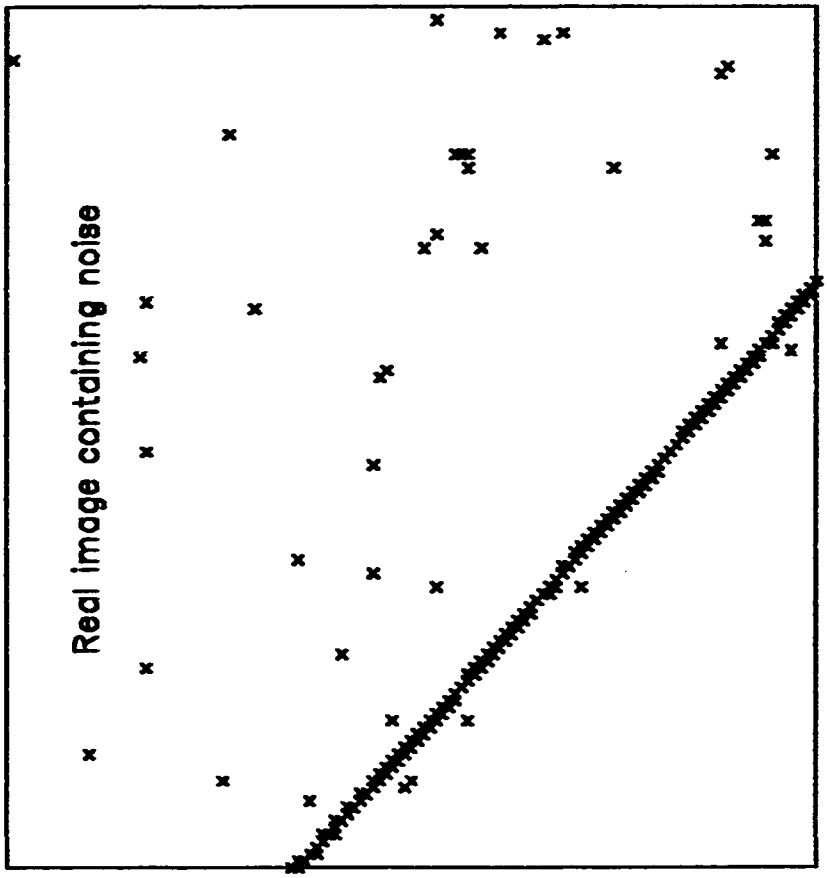


Fig 5.6.1 Real image containing noise

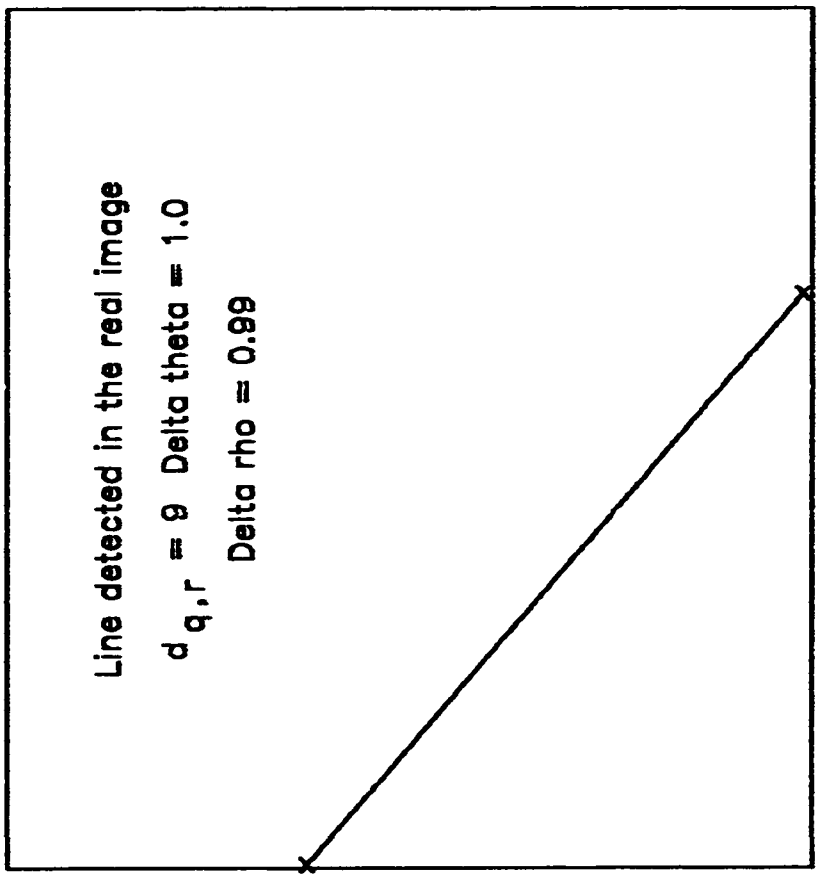


Fig 5.6.2.a Line segment detected from the real image in fig 5.6.1 with $d_{q,r}$ = 9, $\Delta\theta$ = 1.0, $\Delta\rho$ = 0.99, n_{consec} = 2 and η^{k}_{gaps} = 2

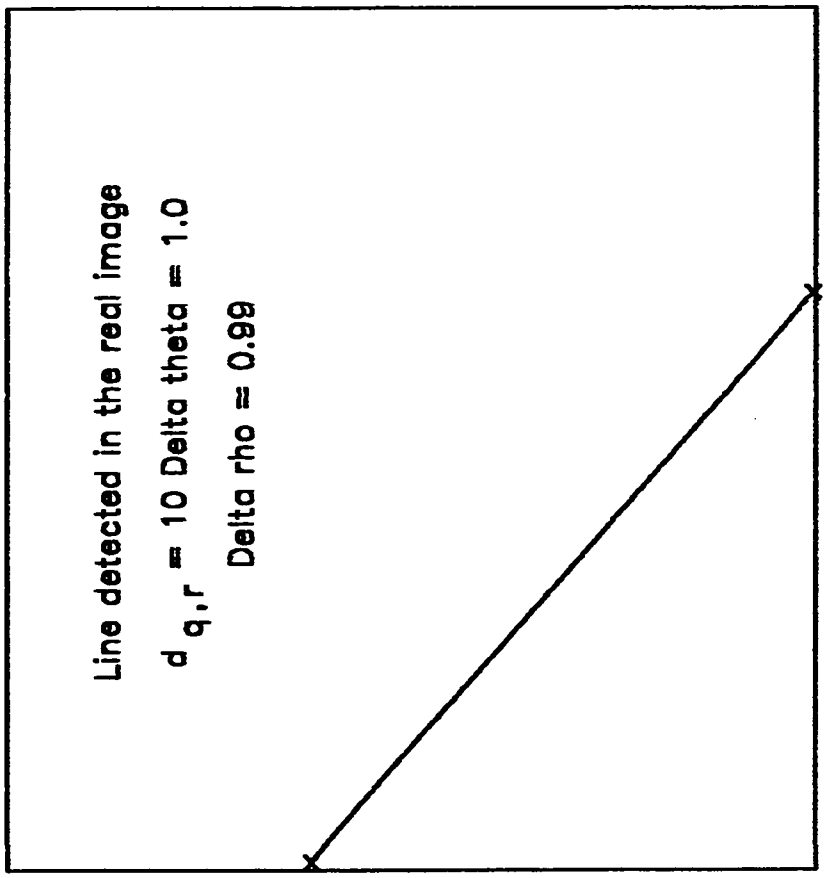


Fig 5.6.2.b Line segment detected from the real image in fig 5.6.1 with $d_{q,r}$ = 10, $\Delta\theta$ = 1.0, $\Delta\rho$ = 0.99, $n_{gaps_{max}}$ = 2 and $\eta^k_{gaps_{max}}$ = 2

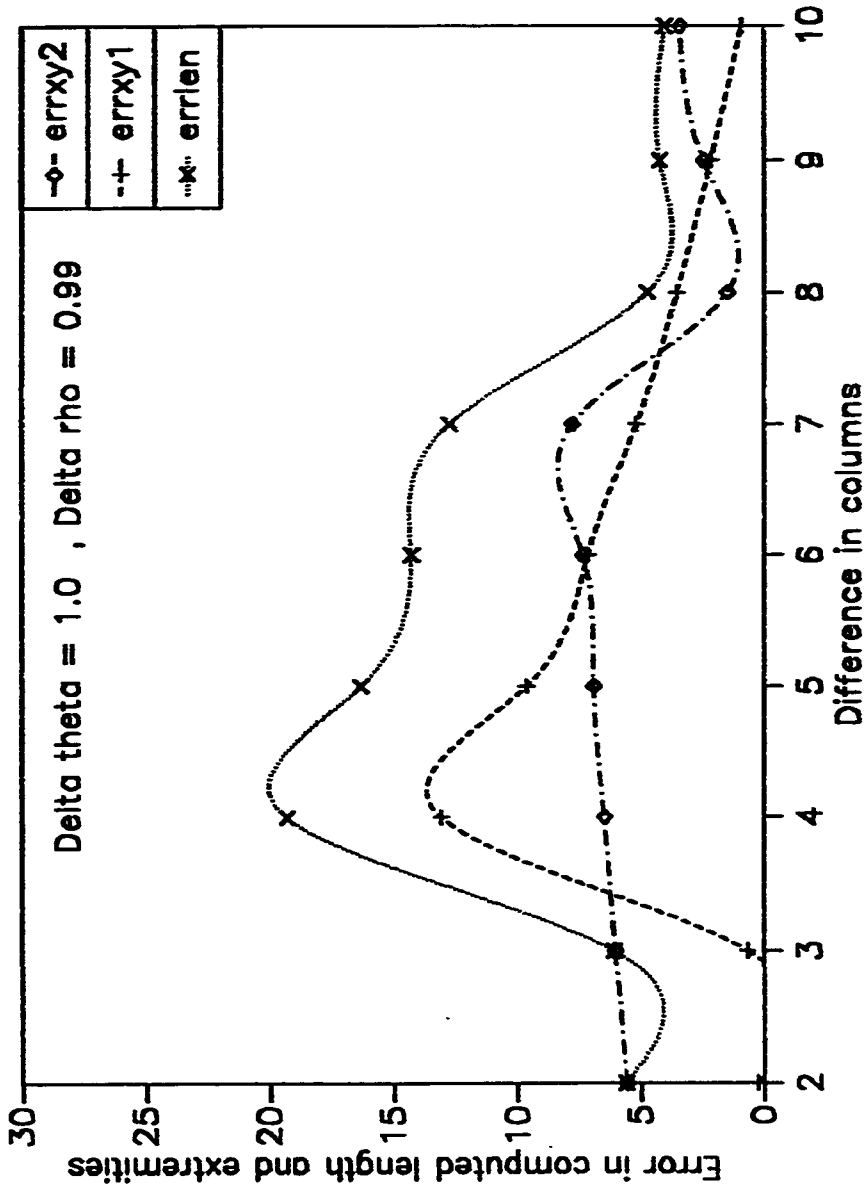


Fig 5.6.3 Variation in error in $l_{computed}$ and computed extremities of the real image in fig 5.6.1 vs $d_{\theta,r}$ for $n_{consec} = 2$ and $\eta^{k}_{correct} = 2$

5.7 Reduction in the Computational Complexity

The computational complexity of the HT depends on the complexities of the step of constructing the accumulator array and the algorithm to find the extremities. The complexity of constructing the accumulator array is dominantly dependent on the value of the resolution $\Delta\theta$. This step has the same order of complexity in all the approaches (the approaches available in literature and the proposed approach). The complexity of this step is $O(N^2)$, where N^2 is the size of the accumulator array. The number of computations required for constructing the accumulator array in the conventional approach are large. This is because, in the conventional HT the primary objective is to determine the values of ρ and θ accurately and to achieve this the resolutions $\Delta\rho$ and $\Delta\theta$ should be small. As $\Delta\theta$ is reduced, the number of sampled values of θ increase, thereby leading to an increase in the number of computations required for constructing the array. In the proposed approach the complexity of this step is reduced by using higher values of $\Delta\theta$. This leads to a significant decrease in the number of computations required in constructing the array.

Let n_f be the number of feature points, n_i be the number of iterations required to determine the extremities and N^2 the size of the accumulator array. In Yamato's [41] approach (sec 4.1) the complexity of the steps of determining the feedback straight line and estimating the number of points lying on the feedback straight line is each of $O(1)$. After this step, the second likely straight line is computed iteratively using the heuristic described in sec 4.1. This step

takes a time of $O(n_f n_i)$. After the second likely straight line is found, the estimated number of feature points are found by projections onto a one-dimensional array which takes $O(n_f)$ time. The scanning of the array for likely endpoints of the line segment is $O(N)$. The complexity of the algorithm is the complexity of the step taking maximum time. In this approach the complexity of the algorithm to determine the extremities of the line segment is $O(n_f n_i + N)$.

Sandler's [42] approach was described in sec 4.2. After the Hough accumulator array is constructed using the combinatorial Hough transform, a one-dimensional histogram is obtained. The size of this one-dimensional histogram depends on the number of pixels (satisfying eq 4.10) and hence the time taken to construct this histogram is $O(n_f)$. Similar to the scanning of the one-dimensional array in Yamato's [41] approach, a connectivity analysis whose complexity is $O(N)$ is performed to determine the extremities. The complexity of the algorithm to find the extremities of the line segment is then $O(n_f + N)$.

In Niblack's [43] approach the principle of surface fitting is used as described in sec 4.3. Computation of the surface to be fitted to the actual Hough accumulator array once takes a time of $O(n_f n_i N)$. The fitting of the surface is continued until the difference between the actual array and the fitted array is less than 0.01. Multiple fittings of the array increase the number of computations. A least square minimization is used to reduce the difference between the actual array and the fitted array. This algorithm has a complexity of $O(N^2)$. Hence the complexity of the algorithm to determine the extremities of

the line segment is $O(n_f n_i N + N^2)$.

The approaches available in literature do not use the information in the accumulator array directly. Techniques such as feedback to the image plane, least square fitting and iterative error minimization are used, which make all these approaches highly computation-intensive as is obvious from the complexity of the algorithms to determine the extremities of the line segment.

In the proposed approach, the extremities of the line segment are computed from two simultaneous equations (eq 4.33 & eq 4.34). These equations are obtained by examining the columns of the accumulator array (as described in sec 4.4) for the first non-zero entry and the last non-zero entry. The solution of the two simultaneous equations requires basic mathematical operations such as addition, multiplication and division which take a constant amount of time. Hence the complexity of the new approach is $O(1)$. This significant decrease in the complexity of the algorithm has been achieved at the cost of a small degradation in accuracy as compared to other methods.

6. CONCLUSIONS

The Hough Transform is a widely used technique for shape detection in computer vision and object recognition. The main drawback of HT is that it requires large storage space and is highly computational. This is because of the large accumulator array, which is required to improve the accuracy of the shape parameters predicted by the HT. On the other side, the information accumulated by the HT has not been fully exploited in literature with respect to obtaining the complete line segment description from the spread in the accumulator array.

Keeping these points in view, the main objectives of this thesis were

- to obtain the complete line segment description from the spread in the accumulator array

and

- to examine the possibilities of what the optimal accumulator size should be.

In the subsequent sections the work done in this thesis is summarized and areas of future work are suggested.

6.1 Summary

The work done in this thesis can be briefly summarized as

1. An extensive literature survey and a thorough understanding of the principle & properties of the Hough Transform for straight line detection.

2. Determination of length of the line segment from the spread in the accumulator array.

3. The expressions to determine the complete line segment description (which includes the extremities and the length together with the normal parameters of the line segment), for *thin* line segments are developed.

4. The expressions for determining the complete line segment description of *thick* line segments are developed.

5. An effort has been made to determine the optimal accumulator array size for the Hough Transform. The approach is based on the fact that, the resolutions $\Delta\rho$ and $\Delta\theta$ be chosen such that the extremities of the line segment are determined accurately and from the computed extremities the value of ρ and θ can be calculated. (In the conventional approach to the HT, $\Delta\rho$ and $\Delta\theta$ are chosen such that ρ

and θ are computed accurately, rather than obtaining the complete line segment description).

6. A reduction is obtained in the computational complexity of the HT by using higher values of $\Delta\theta$ effectively.

7. The new approach has been modified to tackle the problem of noise which is inevitably present in real world images.

6.2 Suggestions for future work

The following areas in the field of Hough Transform are open for further research

1. Determination of the complete line segment description for images containing multiple line segments.

2. Finding the optimum accumulator array size, which could be applied universally (without using an initial estimate of θ).

3. Refinement in the methodology to obtain the complete line segment description in the presence of noise. One way to accomplish this could be the use of variable $\Delta\rho$ i.e., use a value of $\Delta\rho$ which is a function of the sampled value of θ.

4. This thesis has assumed a continuous straight line. The algorithms presented in this thesis can be modified for straight lines having discontinuities.

5. Efficient implementation of the HT on the emerging line of multiprocessors such as the hypercube.

REFERENCES

1. Davies, E.R. 'Machine Vision - Theory, Algorithms, Practicalities', Academic Press, 1990.
2. Duda, R.O., and Hart, P.E., 'Use of the Hough Transformation To Detect Lines and Curves in Pictures', Communications of the ACM, vol 15, No 1, pp 11-15, Jan 1972.
3. Illingworth, J., and Kittler, J., 'A Survey of the Hough Transform', Computer Vision, Graphics and Image Processing, No 44, pp 87-116, 1988.
4. Kimme, C., Ballard, D.H., and Sklansky, J., 'Finding Circles by an Array of Accumulators', Communications of the ACM, No 18, pp 120-122, 1975.
5. Wechsler, H., and Sklansky, J., 'Automatic Detection of Ribs in Chest Radiographs', Pattern Recognition, No 9, pp 21-30, 1977.
6. Tsuji, S., and Matsumoto, F., 'Detection of Ellipses by Modified Hough Transformation', IEEE Transactions on Computers, vol 27, pp 777-781, 1978.
7. Tsukune, H., and Goto, K., 'Extracting Elliptical Figures from an Edge Vector Field', IEEE Conference on Computer Vision and Pattern Recognition, Washington, U.S.A., pp 138-141, 1983.
8. Merlin, P.M., and Farber, D.J., 'A Parallel Mechanism for Detecting Curves in Pictures', IEEE Transactions on Computers, vol 24, pp 96-98, 1975.

9. Sloan, K.R., and Ballard, D.H., 'Experience with the Generalized Hough Transform', The Fifth International Joint Conference on Pattern Recognition, Miami Beach, U.S.A., pp 174-179, 1980.

10. Ballard, D.H., 'Generalizing the Hough Transform to Detect Arbitrary Shapes', Pattern Recognition, No 13, pp 111-122, 1981.

11. Gonzalez, Rafael C., and Wintz Paul, 'Digital Image Processing', Second Edition, Addison-Wesley Publishing Company, 1987.

12. Davies, E.R., 'Image Space Transforms for Detecting Straight Edges in Industrial Images', Pattern Recognition Letters, vol 4, pp 185-192, 1986.

13. Wallace, R.S., 'A Modified Hough Transform for Lines', IEEE Computer Society Conference on Computer Vision and Pattern Recognition, SanFrancisco, U.S.A., pp 665-667, June 19-23, 1985.

14. Brown, C.M., 'Inherent Bias and Noise in the Hough Transform', IEEE Transactions on Pattern Analysis and Machine Intelligence, vol PAMI-5, No 5, pp 493-505, Sept. 1983.

15. Li, H., Lavin, M.A., and LeMaster, R.J., 'Fast Hough Transform: A Hierarchical Approach', Computer Vision, Graphics and Image Processing, No 36, pp 139-161, 1986.

16. Illingworth, J., and Kittler, J., ' The Adaptive Hough Transform', IEEE Transactions on Pattern Analysis and Machine Intelligence, vol PAMI-9, No 5, pp 690-698, Sept. 1987.

17. Princen, J., Yuen, H.K., Illingworth, J., and Kittler, J., 'Properties of the Adaptive Hough Transform', The Sixth Scandinavian Conference on Image Analysis, Qulu, Finland, pp 613-620, June 19-22 1989.

18. Brown, C.M., Curtiss, M.B., and Sher, D.B., 'Advanced Hough Transform Implementations', Eight International Joint Conference on Artificial Intelligence, Karlsruhe, West Germany, pp 1081-1085, Aug 8-12, 1983.

19. Princen, J., Illingworth, J., and Kittler, J., 'A Hierarchical Approach to Line Extraction Based on the Hough Transform', Computer Vision, Graphics and Image Processing, No 52, pp 57-77, 1990.

20. Stephen, R.S., 'Probabilistic Approach to the Hough Transform', Image and Vision Computing, vol 9, No 1, pp 66-71, Feb 1991.

21. Kriyati, N., Eldaar, Y., and Bruckstein, A.M., 'A Probabilistic Hough Transform', Pattern Recognition, vol 24, No 4, pp 303-316, 1991.

22. Shapiro, Stephen D., and Iannino Anthony, 'Geometric Construction for Predicting Hough Transform Performance', IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 1, No 3, pp 310-317, July 1979.

23. Sklansky, J., 'On the Hough Technique for Curve Detection', IEEE Transactions on Computers, vol C-27, No 10, pp 923-926, Oct. 1978.

24. Cohen, M., Toussaint, G.T., 'On the Detection of Structures in Noisy Pictures', Pattern Recognition, vol 9, pp 95-98.

25. Vanveen, T.M. and Groen, F.C.A., 'Discretization Errors in the Hough Transform', Pattern Recognition, vol 14, Nos 1-6, pp 137 - 145, 1981.

26. Srihari, Sagur N. and Govindraju Venugopal, 'Analysis of Textual Images Using the Hough Transform', Machine Vision and Applications, vol 2, pp 141 - 153, 1989.

27. Niblack Wayne, and Petkovic Dragutin, 'On Improving the Accuracy of the Hough Transform - Theory, Simulations and Experiments', IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Ann Arbor, Michigan, U.S.A., pp 574 - 579, June 5-9, 1988.

28. Niblack Wayne, and Petkovic Dragutin, 'On Improving the Accuracy o the Hough Transform', Machine Vision and Applications, vol 3, pp 87 - 106, 1990.

29. Costa L.Da Fontura, and Sandler, M.B., 'Improving Parameter Space for Hough Transform', Electronics Letters, vol 25, No 2, pp 134-136, 19 Jan, 1989.

30. Hanahara, K., Maruyama, T., and Uchiyama, T., 'A Real-Time Processor for the Hough Transform', IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 10, No 1, pp 121-125, 1988.

31. Rhodes, F.M., et-al, 'A Monolithic Hough Transform Processor based on Restructurable VLSII', IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 10, No 1, pp 106-110, 1988.

32. Silberberg, T.M., '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.

33. Li, H., 'Fast Hough Transform for Multidimensional Signal Processing', IBM Research Report RC - 11562, Yorktown Heights, U.S.A, 1985.

34. Little, J.J., Belloch, G., and Cass, T., 'Parallel Algorithms for Computer Vision on the Connection Machine', First IEEE International Conference on Computer Vision, London, U.K, pp 587-591, 1987.

35. Guerra, C., and Hambrusch, S., 'Parallel Algorithms for Line Detection on a Mesh', Journal of Parallel and Distributed Computing, No 6, pp 1-19, 1989.

36. Rosenfield, A., Orleans, J., and Hung, Y., 'Hough Transform Algorithms for Mesh-Connected SIMD Parallel Processors", Computer Vision, Graphics and Image Processing, No 41, pp 293-305,1988.

37. Fisher, A.L.,and Highnam, P.T., 'Computing the Hough Transform on a Scan Line Array Processor', IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 11, No 3, pp 262-265, March 1989.

38. Bowyer, K.W., Jones, J.P., and Lake, C.H., 'Computing the Hough Transform on an MIMD HyperCube', The Sixth Scandinavian Conference on Image Analysis, Qulu, Finland, pp 1172-1181, June 19-22, 1989.

39. Ben-Tzvi, D., Naqvi, A., and Sandler, M.B., 'Synchronous Multiprocessor Implementation of the Hough transform', Computer Vision, Graphics and Image Processing, No 52, pp 437-446, 1990.

40. Skingley, J., and Rye, A.J., 'The Hough Transform Applied to SAR Images for Thin Line Detection', Pattern Recognition Letters, vol 6, pp 61-67, June 1987.

41. Yamato Juniji, Ishii Ikuo, and Makino Hideo, ' Highly Accurate Segment Detection using Hough Transform', Systems and Computers in Japan, vol 21, No 1, pp 68-77, 1990.

42. Sandler, M.B, Costa L.Da Fontura, and Ben-Tzvi Doron, 'Performance Improvements to the Hough Transform', IEE Conference Publication, pp 98-103, March 19-22, 1990.

43. Niblack Wayne and Truong Tai, 'Finding Line Segments by Surface Fitting to the Hough Transform', IAPR International Workshop on Machine Vision Applications, Tokyo, Japan, pp 1-4, Nov 28-30, 1990.

44. Hough, P.V.C., 'A Method and Means for Recognizing Complex Patterns', US Patent 3069654, 1962.

APPENDIX A

This is the pseudo-code of the algorithm discussed in sec 4.5 for determining the extremeties of a line segment in the image plane. From the computed extremeties, the length of the line segment and its normal parameters are determined. The pseudo-code uses the following notation:

Ψ : Set of all feature points in the image space

X , Y : Dimension of the image space

ρ_{max} , θ_{max} : Dimension of the Hough accumulator array

$A(\rho , \theta)$: Accumulator array

ρ_{peak} , θ_{peak} : Cell corresponding to the maximum count in the accumulator array

$j2$: The index in the ρ − direction, corresponding to the first non-zero entry for any θ_k

$j3$: The index in the ρ − direction, corresponding to the last non-zero entry for any θ_k

Begin :

step 1: (* initialization of the hough accumulator array *)

for $k = 0$ to $\theta_{max} - 1$

for $l = 0$ to $\rho_{max} - 1$

$A(k, l) = 0$

endfor

endfor

step 2: (* construction of the hough accumulator array *)

for all $(x, y) \in \Psi$

for $k = 0$ to $\theta_{asize} - 1$

$\theta_k = k \times \Delta\theta$

$\rho_c = x \cos\theta_k + y \sin\theta_k$

if $\rho_c \leq \rho_{max}$

$l = int((\rho_c - \rho_{min})/\Delta\rho)$

$A(k, l) = A(k, l) + 1$

endif

endfor

step 3: (* peak detection and coarse prediction of θ and ρ *)

max = 0

for $i = 0$ to $\theta_{size} - 1$

for $j = 0$ to $\rho_{size} - 1$

if $A(i,j) >$ max

max = $A(i,j)$

$\theta_{peak} = i$

$\rho_{peak} = j$

endif

endfor

endfor

$$\theta_{predicted} = \theta_{min} + i\,\Delta\theta$$

$$\rho_{predicted} = \rho_{min} + j\,\Delta\rho + \frac{\Delta\rho}{2}$$

step 4: (* spread in $A(\theta_{peak}, k)$ *)

$j = 0$

first: if $A(\theta_{peak}, j) = 0$

$j = j + 1$

go to first

else

$j2 = j$

endif

$j = \rho_{size} - 1$

last: if $A(\theta_{peak}, j) = 0$

$j = j - 1$

go to last

else

$j3 = j$

endif

$n_{\rho}^{\theta_{peak}} = j3 - j2 + 1$

step 5: (* computing the extremeties, the length and the normal parameters independent of $\theta_{predicted}$ *)

if $\theta_{predicted} < 45^{\circ}$

$\theta_q > \theta_{peak}$

$\theta_r > \theta_{peak}$

else

$\theta_q < \theta_{peak}$

$\theta_r < \theta_{peak}$

endif

loop

$$\rho_{ql} = \rho_{min} + l_{j2q}\,\Delta\rho + \frac{\Delta\rho}{2}$$

$$\rho_{qh} = \rho_{min} + u_{j3q}\,\Delta\rho + \frac{\Delta\rho}{2}$$

$$\rho_{rl} = \rho_{min} + l_{j2r}\,\Delta\rho + \frac{\Delta\rho}{2}$$

$$\rho_{rh} = \rho_{min} + u_{j3r}\,\Delta\rho + \frac{\Delta\rho}{2}$$

if $\theta_{predicted} < 45^{\circ}$

$$x1c = \frac{\rho_{ql}\sin\theta_r - \rho_{rl}\sin\theta_q}{\sin(\theta_r - \theta_q)}$$

$$y1c = \frac{\rho_{rl}\cos\theta_q - \rho_{ql}\cos\theta_r}{\sin(\theta_r - \theta_q)}$$

$$x2c = \frac{\rho_{qk}\sin\theta_r - \rho_{rk}\sin\theta_q}{\sin(\theta_r - \theta_q)}$$

$$y2c = \frac{\rho_{rk}\cos\theta_q - \rho_{qk}\cos\theta_r}{\sin(\theta_r - \theta_q)}$$

else

$$x1c = \frac{\rho_{qk}\sin\theta_r - \rho_{rk}\sin\theta_q}{\sin(\theta_r - \theta_q)}$$

$$y1c = \frac{\rho_{rk}\cos\theta_q - \rho_{qk}\cos\theta_r}{\sin(\theta_q - \theta_r)}$$

$$x2c = \frac{\rho_{ql}\sin\theta_r - \rho_{rl}\sin\theta_q}{\sin(\theta_r - \theta_q)}$$

$$y2c = \frac{\rho_{rl}\cos\theta_q - \rho_{ql}\cos\theta_r}{\sin(\theta_r - \theta_q)}$$

endif

$$l_{computed} = \sqrt{(x1c - x2c)^2 + (y1c - y2c)^2}$$

$$\theta_{computed} = \tan^{-1}\left(\frac{y2c - y1c}{x2c - x1c}\right) - 90^{\circ}$$

$$\rho_{computed} = \frac{x2c \times y1c - x1c \times y2c}{\sqrt{(x1c - x2c)^2 + (y1c - y2c)^2}}$$

step 6: (* modification to overcome the problem of noise *)

if $\theta_{predicted} > 45^{\circ}$

$$\rho^{q}_{startl} = y_{int} \sin\theta_{q}$$

$$\rho^{q}_{starth} = (N - 1)\cos\theta_{q} + y_{int} \sin\theta_{q}$$

$$\rho^{r}_{startl} = y_{int} \sin\theta_{r}$$

$$\rho^{r}_{starth} = (N - 1)\cos\theta_{r} + y_{int} \sin\theta_{r}$$

else

$$\rho^{q}_{startl} = x_{int} \cos\theta_{q}$$

$$\rho^{q}_{starth} = x_{int} \cos\theta_{q} + (N - 1)\sin\theta_{q}$$

$$\rho^{r}_{startl} = x_{int} \cos\theta_{r}$$

$$\rho^{r}_{starth} = x_{int} \cos\theta_{r} + (N - 1)\sin\theta_{r}$$

$$\eta^{q}_{sl} = int\left(\frac{\rho^{q}_{startl}}{\Delta\rho}\right) - correc$$

$$\eta^{q}_{sh} = int\left(\frac{\rho^{q}_{starth}}{\Delta\rho}\right) - correc$$

$$\eta^r_{sl} = int\left(\frac{\rho^r_{start}}{\Delta\rho}\right) - correc$$

$$\eta^r_{sh} = int\left(\frac{\rho^r_{start}}{\Delta\rho}\right) - correc$$

go to loop in step 5 with

$$l_{j2q} = \eta^q_{sl}$$

$$u_{j3q} = \eta^q_{sh}$$

$$l_{j2r} = \eta^r_{sl}$$

$$u_{j3r} = \eta^r_{sh}$$

End ;

APPENDIX B

```
C ****************************************************************
C   THIS IS THE FORTRAN CODE FOR THE ALGORITHM DEVELOPED TO
C   DETERMINE THE COMPLETE LINE SEGMENT DESCRIPTION OF THE
C   LINE IN THE IMAGE SPACE USING THE HOUGH TRANSFORM.
C ****************************************************************
C
C   IMAGE   :  IMAGE SPACE OF SIZE X * Y CONTAINING A DIGITIZED LINE
C   X       :  DIMENSION OF THE IMAGE SPACE IN THE X-DIRECTION
C   Y       :  DIMENSION OF THE IMAGE SPACE IN THE Y-DIRECTION
C
C   RHO     :  ACTUAL LENGTH OF THE NORMAL FROM ORIGIN TO THE LINE
C              IN THE IMAGE SPACE
C   THETA   :  ACTUAL INCLINATION OF THE NORMAL FROM ORIGIN TO
C              LINE IN THE IMAGE SPACE, WITH RESPECT TO POSITIVE
C              X-AXIS
C   XST &
C   XEND    :  THE LIMITS ON THE X-AXIS BETWEEN WHICH THE DIGITIZED
C              LINE IS GENERATED
C   ROMIN   :  MINIMUM VALUE OF RHO
C   ROMAX   :  MAXIMUM VALUE OF RHO
C   THMIN   :  MINIMUM VALUE OF THETA
C   THMAX   :  MAXIMUM VALUE OF THETA
C
C   HA      :  ACCUMULATOR ARRAY
C   HARO1   :  THE MAXIMUM SIZE OF THE ACCUMULATOR ARRAY IN THE
C              RHO-DIRECTION
C   HARO    :  THE SIZE OF THE ACCUMULATOR ARRAY IN THE
C              RHO-DIRECTION, USED IN COMPUTATION
C   HATH    :  THE SIZE OF THE ACCUMULATOR ARRAY IN THE
C              THETA-DIRECTION
C   DELRO   :  THE RESOLUTION DELTA RHO OF THE ACCUMULATOR ARRAY
C   DELTH   :  THE RESOLUTION DELTA THETA OF THE ACCUMULATOR ARRAY
C
C   C & S   :  LOOK-UP TABLES FOR COSINE AND SINE FUNCTIONS OF
C              ANGLES BETWEEN THMIN AND THMAX
C
C
C ****************************************************************
C
CSSTATEMENTS=0
CSTIME=4000
C
C ****************************************************************
      INTEGER*2  IMAGE(0:127,0:127)
      INTEGER X, Y, THMIN, THMAX, HATH, HARO, HARO1
      INTEGER XST, XEND, YST, YEND
      INTEGER*2 HA(0:127,0:2000)
```

```
      REAL C(0:127), S(0:127), XI(0:127), YI(0:127)
      REAL ERR2(-10:10)
      REAL LENA, LENCA2
      COMMON X,Y
C
C ****************************************************************
C ****************************************************************
C
C   DEFINE THE METHOD OF EXTRAPOLATION TO BE USED IN COMPUTING
C   ENDPOINTS USING THE COARSELY PREDICTED LINE SEGMENT
C        EXTRAP = 0 ---- DIRECT EXTRAPOLATION
C        EXTRAP = 1 ---- EXTRAPOLATION USING AVERAGING
C              INCLUD = 0 ------ EXCLUDE THE CELLS CONTAINING ZEROES
C              INCLUD = 1 ------ INCLUDE THE CELLS CONTAINING ZEROES
C
      EXTRAP = 1
      INCLUD = 0
C
C ****************************************************************
      RHO = 90
      X = 128
      Y = 128
      XST = 69
      XEND = 85
      YST = 30
      YEND = 100
C ****************************************************************
C
C   DEFINE THE NUMBER OF COLUMNS ON EITHER SIDE OF THE PEAK TO
C   TO BE USED WHILE COMPUTING THE AVERAGED LENGTH
C
      J1 = 5
C
C ****************************************************************
      THMIN = 0
      ROMIN = 0.0
      ROMAX = (X-1)*(SQRT (2.0))
      THMAX = 90
      HATH = 121
      HARO1 = 2001
      DELTH = FLOAT(THMAX - THMIN)/FLOAT(HATH - 1)
C     WRITE (11,*) 'IMAGE SIZE = ' , X , '   *   ', Y
C     WRITE (11,*) 'ACTUAL RHO   = ' , RHO
C     WRITE (11,*) 'HA THETA SIZE = ' , HATH
C     WRITE (11,*) 'HA RHO  SIZE  = ' , HARO
C     WRITE (11,*) 'DELTA THETA   = ' , DELTH
C     WRITE (11,*) 'DELTA RHO     = ' , DELRO
C     WRITE(11,*) '
C     WRITE(11,4) (K2, K2 = -10,10)
C4    FORMAT(6X,21(I3,3X))
C     WRITE(11,*) '
C     WRITE(11,6)
```

```
C6      FORMAT(1X,'THETA',2X,'BAR1',2X,'BAR2',6X,'X1C',6X,'Y1C',
C     * 6X,'X2C',6X,'Y2C',5X,'LENC')
        DO 10 M5 = 45,45
         THETA = M5
         WRITE (6,*) ' THETA = ', THETA
C ***************************************************************
C ***************************************************************
        WRITE (10,*) 'IMAGE SIZE = ' , X , '   *   ', Y
        WRITE (10,*) 'ACTUAL THETA = ' , THETA
        WRITE (10,*) 'ACTUAL RHO   = ' , RHO
        WRITE (10,*) 'HA THETA SIZE = ' , HATH
        WRITE (10,*) 'HA RHO  SIZE  = ' , HARO
        WRITE (10,*) 'DELTA THETA   = ' , DELTH
        WRITE (10,*) 'DELTA RHO     = ' , DELRO
C ***************************************************************
        CALL INIT (IMAGE, X, Y)
C ***************************************************************
C
        CALL LINE (RHO, THETA, X, Y, X1, Y1, XST, XEND, YST, YEND,IMAGE)
C ***************************************************************
        CALL RESULT (IMAGE, X, Y, THETA)
C ***************************************************************
        CALL ACTUAL (IMAGE, X, Y, THETA, X1A, Y1A, X2A, Y2A)
C ***************************************************************
C       DO 8 I21 = 600, 1200, 200
        DO 8 I21 = 150, 150
C ***************************************************************
             HARO = I21
             DELRO = (ROMAX - ROMIN)/FLOAT(HARO - 1)
C ***************************************************************
        CALL INITI (HA, ROMIN, ROMAX, THMIN, THMAX, HARO1,
     * HARO, HATH)
C ***************************************************************
        CALL CI (THMIN, THMAX, HATH, DELTH, C, S)
C ***************************************************************
        CALL HOUGH (IMAGE, ROMIN,ROMAX, THMIN, THMAX, HARO1,
     * HARO, HATH, X, Y, C, S, DELTH, DELRO, HA)
C ***************************************************************
        CALL MAXIJ (HA, THMIN, THMAX, ROMIN, ROMAX, HATH, HARO1,
     * HARO, DELTH, DELRO, THP, ROP, MTH, MRO)
C ***************************************************************
        CALL RESULTI (HA, THMIN, THMAX, ROMIN, ROMAX, HATH, HARO1,
     * HARO, DELTH, DELRO, THP, ROP, MTH, MRO)
C ***************************************************************
        CALL SPREAD (HA, THMIN, THMAX, ROMIN, ROMAX, HATH, HARO1,
     * HARO, THP, ROP, DELTH, DELRO, MTH, MRO, NCOUNT)
C ***************************************************************
        CALL AVERAGE (HA, THMIN, THMAX, ROMIN, ROMAX, HATH, HARO1,
     * HARO, DELTH, DELRO, THP, ROP, J1, MTH, MRO, AVL)
C ***************************************************************
        CALL THICK (HA, THMIN, THMAX, ROMIN, ROMAX, HATH, HARO1,
     * HARO, DELTH, DELRO, THP, ROP, MTH, MRO, AVL, NCOUNT, J1,
```

```
     * WIDTH)
C ****************************************************************
C       CALL COORD (THETA, HA, THMIN, THMAX, ROMIN, ROMAX,
C     * HATH, HARO1, HARO, DELTH, DELRO, THP, ROP, MTH, MRO,
C     * X1A, Y1A, X2A, Y2A, EXTRAP, INCLUO, ERR2)
C ****************************************************************
C ****************************************************************
        DO 7 I22 = 10, 10
           IF(THP.LT.45.0) THEN
              IBAR1 = MTH + 20
              IBAR2 = IBAR1 + I22
           ELSE
              IBAR1 = MTH - 20
              IBAR2 = IBAR1 - I22
           ENDIF
C ****************************************************************
        CALL COORD2 (THETA, HA, THMIN, THMAX, ROMIN, ROMAX, HATH,
      * HARO1, HARO, DELTH, DELRO, THP, ROP, MTH, MRO,
      * X1A, Y1A, X2A, Y2A, IBAR1, IBAR2, LENA, WIDTH)
C ****************************************************************
7       CONTINUE
8       CONTINUE
C       WRITE (10,*) 'PREDICTED THETA      = ' , THP
C       WRITE (10,*) 'PREDICTED RHO        = ' , ROP
C       WRITE (10,*) 'CELL THETA PREDICTED = ' , MTH
C       WRITE (10,*) 'CELL RHO PREDICTED   = ' , MRO
C       WRITE (10,*) ' NO: OF CELLS USED IN LENGTH COMPUTATION = ', 2*J1
C       WRITE (10,*) 'AVERAGE LENGTH       = ' , AVL
C       WRITE (11,8) THETA, (ERR2(K2), K2 = -10,10)
C8      FORMAT (1X,F4.1,21(F4.1,2X))
C       WRITE(11,*) '
C       WRITE (11,*) 'X1A = ', X1A, ' Y1A = ', Y1A
C       WRITE (11,*) 'X2A = ', X2A, ' Y2A = ', Y2A
C       WRITE (11,*) ' ACTUAL LENGTH = ', LENA
C       WRITE(11,*) '
C       WRITE(11,*) '
C       WRITE(11,*) '
C ****************************************************************
10      CONTINUE
        STOP
        END
C ****************************************************************
C         THIS SUBROUTINE INTIALIZES THE ARRAY IMAGE
C ****************************************************************
        SUBROUTINE INIT (IMAGE, X, Y)
        INTEGER X, Y
        INTEGER*2  IMAGE(0:X-1,0:Y-1)
        DO 20 I = 0, X-1
          DO 15 J = 0, Y-1
            IMAGE(I,J) = 0
15        CONTINUE
20      CONTINUE
```

```
      RETURN
      END
C ***************************************************************
C  THIS SUBROUTINE GENERATES A LINE FOR A GIVEN VALUE OF RHO AND
C  THETA; THE GENERATED LINE IS STORED IN THE ARRAY IMAGE.
C ***************************************************************
      SUBROUTINE LINE (RHO, THETA, X, Y, X1, Y1, XST, XEND,
     * YST, YEND, IMAGE)
      INTEGER X, Y
      INTEGER XST, XEND, YST, YEND
      INTEGER*2  IMAGE(0:X-1,0:Y-1)
      REAL X1(0:X-1), Y1(0:Y-1)
      PI = 3.142857143
      RAD = (THETA)*(PI/180.0)
      DO 25 I = XST, XEND
      Y1(I) = (RHO)/SIN(RAD) - I*COS(RAD)/SIN(RAD)
      Z = Y1(I)
      K = INT(Z)
      IF (K.GE.0) THEN
         IF(K.LE.Y-1) THEN
           IMAGE(I,K) = 1
         ENDIF
      ENDIF
25    CONTINUE
      DO 30 J = 0,Y-1
      X1(J) = (RHO)/COS(RAD) - J*TAN(RAD)
      W = X1(J)
      L = INT(W)
      IF (L.GE.XST) THEN
         IF(L.LE.XEND) THEN
           IMAGE(L,J) = 1
         ENDIF
      ENDIF
30    CONTINUE
      RETURN
      END
C ***************************************************************
C  THIS SUBROUTINE STORES THE CO-ORDINATES OF THE IMAGE PLANE
C  CORRESPONDING TO THE POINTS OF THE DIGITIZED LINE
C ***************************************************************
      SUBROUTINE RESULT (IMAGE, X, Y, THETA)
      INTEGER X, Y
      INTEGER*2  IMAGE(0:X-1,0:Y-1)
      KSUM = 0
      DO 40 I = 0, X-1
         DO 35 J = 0, Y-1
           IF (IMAGE(I,J).EQ.1) THEN
           WRITE (7,*) I,J
           KSUM = KSUM + 1
           ENDIF
35       CONTINUE
40    CONTINUE
```

```
      WRITE (6,*) 'NUMBER OF POINTS ON THE IMAGE LINE = ' , KS
      RETURN
      END
C ***************************************************************
C ***************************************************************
C    THIS SUBROUTINE READS THE ACTUAL ENDPOINTS OF THE LINE
C    SEGMENT IN THE IMAGE SPACE
C ***************************************************************
      SUBROUTINE ACTUAL (IMAGE, X, Y, THETA, X1A, Y1A, X2A, Y2
      INTEGER X, Y
      INTEGER*2  IMAGE(0:X-1,0:Y-1)
      IF (THETA.LT.45.0) THEN
         DO 43 K1 = 0,X-1
            DO 43 L1 = Y-1,0,-1
               IF(IMAGE(K1,L1).EQ.1) GO TO 44
43       CONTINUE
44       X2A = K1
         Y2A = L1
      ELSE
         DO 45 L1 = 0,Y-1
            DO 45 K1 = X-1,0,-1
               IF(IMAGE(K1,L1).EQ.1) GO TO 46
45       CONTINUE
46       X1A = K1
         Y1A = L1
      ENDIF
      IF (THETA.LT.45) THEN
         DO 47 K1 = X-1,0,-1
            DO 47 L1 = 0,Y-1
              IF(IMAGE(K1,L1).EQ.1) GO TO 48
47       CONTINUE
48       X1A = K1
         Y1A = L1
      ELSE
         DO 49 L1 = Y-1,0,-1
            DO 49 K1 = 0,X-1
              IF(IMAGE(K1,L1).EQ.1) GO TO 50
49       CONTINUE
50       X2A = K1
         Y2A = L1
      ENDIF
      WRITE(6,*) 'X1A = ', X1A, ' Y1A = ', Y1A
      WRITE(6,*) 'X2A = ', X2A, ' Y2A = ', Y2A
      RETURN
      END
C ***************************************************************
C ***************************************************************
C  THIS SUBROUTINE INTIALIZES THE HOUGH ACCUMULATOR ARRAY HA
C ***************************************************************
      SUBROUTINE INITI (HA, ROMIN, ROMAX, THMIN, THMAX,
     * HARO1, HARO, HATH)
      INTEGER THMIN, THMAX, HARO, HATH, HARO1
```

```
      INTEGER*2 HA(0:HATH-1, 0:HARO1-1)
      DO 90 K = 0, HATH-1
        DO 85 L = 0, HARO-1
          HA(K,L) = 0
85      CONTINUE
90    CONTINUE
      RETURN
      END
C ****************************************************************
C  THIS SUBROUTINE COMPUTES THE COSINE AND SINE OF ALL ANGLES
C  BETWEEN THMIN AND THMAX
C ****************************************************************
      SUBROUTINE C1 (THMIN, THMAX, HATH, DELTH, C, S)
      INTEGER THMIN, THMAX, HATH
      REAL C(0:HATH-1), S(0:HATH-1)
      PI = 3.142857143
      DO 95 K = 0, HATH-1
        TH1 = K * DELTH
        RTH1 = TH1 * ((PI)/180.0)
        C(K) = COS(RTH1)
        S(K) = SIN(RTH1)
95    CONTINUE
      RETURN
      END
C ****************************************************************
C  THIS SUBROUTINE ACCUMULATES THE HOUGH ACCUMULATOR ARRAY.
C ****************************************************************
      SUBROUTINE HOUGH (IMAGE, ROMIN,ROMAX, THMIN, THMAX, HARO1
     * HARO, HATH, X, Y, C, S, DELTH, DELRO, HA)
      INTEGER X, Y, THMIN, THMAX, HARO, HATH, HARO1
      INTEGER*2 HA(0:HATH-1, 0:HARO1-1)
      INTEGER*2  IMAGE(0:X-1,0:Y-1)
      REAL C(0:HATH-1), S(0:HATH-1)
      DO 110 K = 0, X-1
        DO 105 L = 0, Y-1
          IF (IMAGE(K,L).EQ.1) THEN
            DO 100 I = 0, HATH-1
              RHOC =(K * (C(I))) + (L * (S(I)))
              IF ((RHOC).LE.(ROMAX)) THEN
                ROD = (RHOC - ROMIN)/(DELRO)
                J = INT (ABS(ROD))
                HA(I,J) = HA(I,J) + 1
              ENDIF
100         CONTINUE
          ENDIF
105     CONTINUE
110   CONTINUE
      RETURN
      END
C ****************************************************************
C ****************************************************************
C  THIS SUBROUTINE PREDICTS THE RHO AND THETA OF THE LINE BY FINDING
```

```
C  THE CELL IN THE ACCUMULATOR ARRAY HAVING THE MAXIMUM NUMBER OF
C  VOTES
C ******************************************************************
      SUBROUTINE MAXIJ (HA, THMIN, THMAX, ROMIN, ROMAX, HATH,
     * HARO1, HARO, DELTH, DELRO, THP, ROP, MTH, MRO)
      INTEGER THMIN, THMAX, HARO, HATH, HARO1
      INTEGER*2 HA(0:HATH-1,0:HARO1-1)
      AMAX = -1.0E+20
      DO 120 I = 0, HATH-1
        DO 115 J = 0, HARO-1
          IF (HA(I,J).GT.AMAX) THEN
            AMAX = HA(I,J)
            THCELL = I
            ROCELL = J
          ENDIF
115       CONTINUE
120     CONTINUE
        MTH = THCELL
        MRO = ROCELL
      THP = THMIN + MTH * DELTH
      ROP = ROMIN + MRO * DELRO + DELRO/2.0
      RETURN
      END
C ******************************************************************
C  THIS SUBROUTINE STORES THE ACCUMULATOR ARRAY
C ******************************************************************
      SUBROUTINE RESULT1 (HA, THMIN, THMAX, ROMIN, ROMAX, HATH
     *  HARO1, HARO, DELTH, DELRO, THP, ROP, MTH, MRO)
      INTEGER THMIN, THMAX, HARO, HATH, HARO1
      INTEGER*2 HA(0:HATH-1, 0:HARO1-1)
      DO 125 J = MRO + 50, MRO - 50, -1
        WRITE (10,130) J, (HA(I,J),I= MTH - 10, MTH + 10)
125     CONTINUE
        WRITE (10,135) (I, I= MTH - 10, MTH + 10)
130     FORMAT (1X, I4, 1X, 21I3)
135     FORMAT (6X, 21I3)
      RETURN
      END
C ******************************************************************
C  THIS SUBROUTINE DETERMINES THE SPREAD CORRESPONDING TO THE PEAK
C  IN THE ACCUMULATOR ARRAY.
C ******************************************************************
      SUBROUTINE SPREAD (HA, THMIN, THMAX, ROMIN, ROMAX, HATH,HARO1,
     * HARO, THP, ROP, DELTH, DELRO, MTH, MRO, NCOUNT)
      INTEGER THMIN, THMAX, HARO, HATH, HARO1
      INTEGER*2 HA(0:HATH-1,0:HARO1-1)
      J = 0
160   IF (HA(MTH,J).EQ.0) THEN
        J = J+1
        GO TO 160
      ELSE
        J2 = J
```

```
      ENDIF
      J = HARO-1
170   IF (HA(MTH,J).EQ.0) THEN
         J = J-1
         GO TO 170
      ELSE
         J3 = J
      ENDIF
C     WRITE (6,*) 'J2 = ' , J2, '   J3 = ' , J3
      NCOUNT = J3-J2+1
C     WRITE (6,*) 'SPREAD IN HA AT PREDICTED THETA = ' , NCOUNT
      WRITE (10,*) 'SPREAD IN HA AT PREDICTED THETA = ' , NCOUNT
      RETURN
      END
C *****************************************************************
C  THIS SUBROUTINE CALCULATES THE AVERAGE LENGTH OF THE LINE FROM
C  THE SPREAD IN THE ACCUMULATOR ARRAY
C *****************************************************************
      SUBROUTINE AVERAGE (HA, THMIN, THMAX, ROMIN, ROMAX, HATH,
     * HARO1, HARO, DELTH, DELRO, THP, ROP, J1, MTH, MRO, AVL)
      INTEGER THMIN, THMAX, HARO, HATH, HARO1
      INTEGER*2 HA(0:HATH-1,0:HARO1-1)
      AVL1 = 0.0
      PI = 3.142857143
      DO 200 I = -J1, J1
         IF (I.NE.0) THEN
            RAD1 = ((PI)/180.0) * ((DELTH)/2 + (ABS(I) * (DELTH)))
            SIN1 = SIN(RAD1)
            K = MTH + I
            J = 0
173         IF (HA(K,J).EQ.0) THEN
               J = J+1
               IF (J.LE.HARO-1) THEN
                  GO TO 173
               ENDIF
            ELSE
               J2 = J
            ENDIF
            J = HARO-1
176         IF (HA(K,J).EQ.0) THEN
               J = J-1
               IF (J.NE.0) THEN
                  GO TO 176
               ENDIF
            ELSE
               J3 = J
               ENDIF
            ICOUNT = J3-J2+1
            AVL1 = AVL1 + ((((ICOUNT -1) * DELRO)/SIN1)
         ENDIF
200   CONTINUE
      AVL = AVL1/(2 * J1)
```

```
      RETURN
      END
C *****************************************************************
C *****************************************************************
C   THIS SUBROUTINE DETERMINES THE COCORDINATES OF THE LINE FROM THE
C   SPREAD IN THE ACCUMULATOR ARRAY AND THE COARSELY PREDICTED
C   LINE SEGMENT
C *****************************************************************
      SUBROUTINE COORD (THETA, HA, THMIN, THMAX, ROMIN, ROMAX,
     * HATH, HARO1, HARO, DELTH, DELRO, THP, ROP, MTH, MRO,
     * X1A, Y1A, X2A, Y2A, EXTRAP, INCLUO, ERR2)
      INTEGER THMIN, THMAX, HARO, HATH, HARO1
      INTEGER*2 HA(0:HATH-1,0:HARO1-1)
      REAL LENA, LENC, LENE
      REAL ERR2(-10:10)
      DO 201 K3 = 10,10
       ERR2(K3) = 0.0
201   CONTINUE
      PI = 3.142857143
      LENA = SQRT((X1A - X2A)**2 + (Y1A - Y2A)**2)
      WRITE (10,*) '
      WRITE (10,*) 'X1 = ',X1A,'  Y1 = ',Y1A,' X2 = ',X2A,'Y2=',Y2A
      WRITE (10,*) '
      WRITE (10,*) 'ACTUAL LENGTH = ', LENA
      IF (EXTRAP.EQ.0) THEN
         WRITE (10,*) '
         WRITE (10,*) ' DIRECT EXTRAPOLATION USED'
      ELSEIF (INCLUO.EQ.0) THEN
         WRITE (10,*) '
         WRITE (10,*) ' EXTRAPOLATION USING AVERAGING WITH ZEROES
     *    EXCLUDED'
      ELSE
         WRITE (10,*) '
         WRITE (10,*) ' EXTRAPOLATION USING AVERAGING WITH ZEROES
     *    INCLUDED'
      ENDIF
      WRITE (10,*) '
      WRITE (10,202)
202   FORMAT('IBAR',5X,'X1C',7X,'Y1C',7X,'X2C',7X,'Y2C',6X,'LENC',
     * 7X,'X1E',7X,'Y1E',7X,'X2E',7X,'Y2E',6X,'LENE')
C     I5 = -10
C     I6 = -6
205   DO 300 N1 = -10, 10
      IF (N1.NE.0) THEN
      JX1 = 0
210   IF (HA((MTH+N1),JX1).EQ.0) THEN
         JX1 = JX1 + 1
C        IF (JX1.LE.HARO-1) THEN
            GO TO 210
C        ENDIF
      ELSE
         ROMN1 = JX1
```

```
        ENDIF
        JX2 = HARO-1
 220    IF (HA((MTH+N1),JX2).EQ.0) THEN
           JX2 = JX2-1
 C         IF (JX2.NE.0) THEN
               GO TO 220
 C         ENDIF
        ELSE
           ROMX1 = JX2
        ENDIF
        ROMN = ROMIN + (ROMN1) * (DELRO) + DELRO/2.0
        ROMX = ROMIN + (ROMX1) * (DELRO) + DELRO/2.0
        STHP = SIN(THP * PI/180.0)
        CTHP = COS(THP * PI/180.0)
        TH1 = THP + N1*DELTH
        STH1 = SIN(TH1 * PI/180.0)
        STH2 = SIN(N1*DELTH * PI/180.0)
 C      STH2 = SIN(ABS(N1*DELTH * PI/180.0))
 C      WRITE(6,*) N1, STH2
        IF (N1.GT.0) THEN
           X1C = ((ROP * STH1) - (ROMN * STHP))/STH2
           X2C = ((ROP * STH1) - (ROMX * STHP))/STH2
        ELSE
           X1C = ((ROP * STH1) - (ROMX * STHP))/STH2
           X2C = ((ROP * STH1) - (ROMN * STHP))/STH2
        ENDIF
        Y1C = (ROP - (X1C * CTHP))/STHP
        Y2C = (ROP - (X2C * CTHP))/STHP
        LENC = SQRT((X1C - X2C)**2 + (Y1C - Y2C)**2)
        ERRL1C = SQRT((X1A - X1C)**2 + (Y1A - Y1C)**2)
        ERRL2C = SQRT((X2A - X2C)**2 + (Y2A - Y2C)**2)
        ERRLENC = ((LENA - LENC)/LENA)*100
 C ***************************************************************
 C                    EXTRAPOLATION
 C ***************************************************************
        TH3 = ((DELTH/2.0) + N1 * DELTH) * PI/180.0
        ROLEN1 = DELRO/SIN(ABS(TH3))
        BIN1 = HA(MTH+N1,JX1)
        BIN2 = HA(MTH+N1,JX2)
        JX3 = JX1 + 1
        JX4 = JX2 - 1
        IF (EXTRAP.EQ.0) THEN
         BIN3 = HA(MTH+N1,JX3)
         BIN4 = HA(MTH+N1,JX4)
 C       WRITE (6,*) 'N1 = ',N1
 C       WRITE (6,*) 'BIN1 = ',BIN1, ' BIN2 = ',BIN2
 C       WRITE (6,*) 'BIN3 = ',BIN3, ' BIN4 = ',BIN4
        ELSE
         CNTR = 0
         VOTE = 0
         DO 250 I1 = JX1+1,JX2-1
            VOTE = VOTE + HA(MTH+N1,I1)
```

```
            IF (HA(MTH+N1,I1).NE.0) THEN
               CNTR = CNTR + 1
            ENDIF
250      CONTINUE
         IF (INCLUD.EQ.0) THEN
            BIN3 = VOTE/CNTR
C           WRITE (6,*) 'NON ZERO CELLS = ', CNTR
         ELSE
            BIN3 = VOTE/(JX2-JX1-1)
C           WRITE (6,*) 'TOTAL CELLS = ', JX2-JX1-1
         ENDIF
         BIN4 = BIN3
        ENDIF
        IF (N1.GT.0) THEN
           ROLEN2 = (BIN1/BIN3) * ROLEN1
           ROLEN3 = (BIN2/BIN4) * ROLEN1
           X3 = ((ROP * STH1) - ((ROMIN + (JX3 * DELRO))*STHP))/STH2
           X4 = ((ROP * STH1) - ((ROMIN + (JX2 * DELRO))*STHP))/STH2
        ELSE
           ROLEN2 = (BIN2/BIN4) * ROLEN1
           ROLEN3 = (BIN1/BIN3) * ROLEN1
           X3 = ((ROP * STH1) - ((ROMIN + (JX2 * DELRO))*STHP))/STH2
           X4 = ((ROP * STH1) - ((ROMIN + (JX3 * DELRO))*STHP))/STH2
        ENDIF
        Y3 = (ROP - (X3 * CTHP))/STHP
        Y4 = (ROP - (X4 * CTHP))/STHP
        TH4 = THP
C       TH4 = THP + DELTH/2.0
        RTH4 = TH4 * PI/180.0
        X1E = X3 + (SIN(RTH4)*ROLEN2)
        Y1E = Y3 - (COS(RTH4)*ROLEN2)
        X2E = X4 - (SIN(RTH4)*ROLEN2)
        Y2E = Y4 + (COS(RTH4)*ROLEN2)
        LENE = SQRT((X1E - X2E)**2 + (Y1E - Y2E)**2)
        ERRL1E = SQRT((X1A - X1E)**2 + (Y1A - Y1E)**2)
        ERRL2E = SQRT((X2A - X2E)**2 + (Y2A - Y2E)**2)
        ERRLENE = ((LENA - LENE)/LENA)*100
        ERR2(N1) = ABS(ERRLENE)
C       WRITE (10,*) ' BAR IN THETA DIRECTION = ' , MTH + N1
C       WRITE (10,*) 'X1 = ' , X1 , '  Y1 = ', Y1
C       WRITE (10,*) 'X2 = ' , X2 , '  Y2 = ', Y2
        IBAR = MTH + N1
        WRITE (10,*) '
        WRITE (10,295) IBAR,X1C, Y1C, X2C, Y2C, LENC,
     * X1E, Y1E, X2E, Y2E, LENE
        WRITE (10,298) ERRL1C, ERRL2C, ERRLENC, ERRL1E, ERRL2E,
     * ERRLENE
295     FORMAT(I3,10(3X,F7.2))
298     FORMAT(11X,F7.3,13X,F7.3,8X,F7.3,8X,F7.3,13X,F7.3,8X,F7.
C       WRITE (10,*) 'X2 = ' , X2 , '  Y2 = ', Y2
        ENDIF
300     CONTINUE
```

```
C       I5 = 6
C       I6 = 10
C       GOTO 205
        RETURN
        END
C*****************************************************************
C*****************************************************************
C   THIS SUBROUTINE DETERMINES THE NUMBER OF FEATURE POINTS IN THE
C   IMAGE SPACE, HAVING ONE OF THE CO-ORDINATES EQUAL
C
C*****************************************************************
        SUBROUTINE THICK (HA, THMIN, THMAX, ROMIN, ROMAX, HATH,HARO1,
     * HARO, DELTH, DELRO, THP, ROP, MTH, MRO, AVL, NCOUNT, J1,
     * WIDTH)
        INTEGER THMIN, THMAX, HARO, HATH, HARO1
        INTEGER*2 HA(0:HATH-1,0:HARO1-1)
        PI = 3.142857143
C       WRITE (6,*) 'J1 = ', J1
C       WRITE (6,*) 'NCOUNT = ', NCOUNT, ' AVL = ', AVL
C       SLOPE = ABS((COS(THP * PI/180.0))/(SIN(THP * PI/180.0)))
        SLOPE = -(COS(THP * PI/180.0))/(SIN(THP * PI/180.0))
C       POINTS = ABS(1.0/SLOPE)
        POINTS = ABS((1.0/SLOPE)) + 1
        WIDTH = (POINTS) * (COS(THP * PI/180.0))
        ALEN = (NCOUNT * DELRO)/(SIN(DELTH/2.0 * PI/180.0))
C       WRITE (6,*) 'LENGTH FROM PEAK = ', ALEN
C       DELTA = ASIN(WIDTH/AVL)
C       DELTA = ASIN(WIDTH/ALEN)
C       DELTA = (DELTA * 180.0/PI)
C       DELTA = ATAN(WIDTH/AVL)
        DELTA = ATAN(WIDTH/ALEN)
        DELTA = (DELTA * 180.0/PI)
        AVL2 = 0.0
        DO 350 I = -J1, J1
           IF (I.NE.0) THEN
              RAD2=((PI)/180.0)*(DELTA+(DELTH)/2+(ABS(I)*(DELTH)))
              SIN2 = SIN(RAD2)
              K = MTH + I
              J = 0
320           IF (HA(K,J).EQ.0) THEN
                 J = J+1
                 IF (J.LE.HARO-1) THEN
                    GO TO 320
                 ENDIF
              ELSE
                 J2 = J
              ENDIF
              J = HARO-1
330           IF (HA(K,J).EQ.0) THEN
                 J = J-1
                 IF (J.NE.0) THEN
                    GO TO 330
```

```
                ENDIF
              ELSE
                J3 = J
                ENDIF
              ICOUNT1 = J3-J2+1
              AVL2 = AVL2 + (((ICOUNT1 -1) * DELRO)/SIN2)
            ENDIF
350     CONTINUE
        AVL2 = AVL2/(2 * J1)
        WRITE (6,*) 'NUMBER OF POINTS HAVING SAME Y-COORD = ', POINTS
        WRITE (6,*) 'DELTA = ', DELTA
C       WRITE (6,*) 'AVERAGE LENGTH  OF THE THICK LINE = ', AVL2
        RETURN
        END
C*****************************************************************
C*****************************************************************
C  THIS SUBROUTINE DETERMINES THE EXTREMITIES OF THE LINE SEGMENT
C  FROM THE SPREAD IN THE ACCUMULATOR ARRAY INDEPENDENT OF THE
C  PREDICTED LINE SEGMENT
C*****************************************************************
        SUBROUTINE COORD2 (THETA, HA, THMIN, THMAX, ROMIN, ROMAX,
     * HATH, HARO1, HARO, DELTH, DELRO, THP, ROP, MTH, MRO, X1A,Y1A
     * X2A, Y2A, IBAR1, IBAR2, LENA, WIDTH,TR, ERRL, ERRXY1, ERRXY2)
        INTEGER THMIN, THMAX, HARO, HATH, HARO1
        INTEGER*2 HA(0:HATH-1,0:HARO1-1)
        REAL LENA, LENCA2
        REAL ERRL(40) , ERRXY1(40) , ERRXY2(40)
        PI = 3.142857143
        LENA = SQRT((X1A - X2A)**2 + (Y1A - Y2A)**2)
        ICONSECL = 3
        ICONSECH = 3
        ICORRL = 2
        ICORRH = 2
        ICORRL1 = 2
        ICORRH1 = 2
        THET1 = THMIN + IBAR1 * DELTH
        THET2 = THMIN + IBAR2 * DELTH
        DTH12 = THET2 - THET1
        RTHP = (PI * THP)/180.0
        RTHET1 = (PI * THET1)/180.0
        RTHET2 = (PI * THET2)/180.0
        RDTH12 = (PI * DTH12)/180.0
        SIN1A2 = SIN(RTHET1)
        COS1A2 = COS(RTHET1)
        SIN2A2 = SIN(RTHET2)
        COS2A2 = COS(RTHET2)
        SINDTH = SIN(RDTH12)
C       MJ1L = 0
C       MJ1H = HARO-1
C       MJ2L = 0
C       MJ2H = HARO - 1
        IF(THP.LT.45.0) THEN
```

```
        MJSL = 0
        CALL RHOSL1(RTHP,ROP,RTHET1,ICORRL1,DELRO,MJSL)
        MJ1L = MJSL
        RO1L = ROMIN + (MJ1L * DELRO) + (DELRO/2.0)
        MJSL = 0
        CALL RHOSL1(RTHP,ROP,RTHET2,ICORRL1,DELRO,MJSL)
        MJ2L = MJSL
        RO2L = ROMIN + (MJ2L * DELRO) + (DELRO/2.0)
        MJSH = 0
        CALL RHOSH1(RTHP,ROP,RTHET1,ICORRH1,DELRO,MJSH)
        MJ1H = MJSH
        RO1H = ROMIN + (MJ1H * DELRO) + (DELRO/2.0)
        MJSH = 0
        CALL RHOSH1(RTHP,ROP,RTHET2,ICORRH1,DELRO,MJSH)
        MJ2H = MJSH
        RO2H = ROMIN + (MJ2H * DELRO) + (DELRO/2.0)
C
       ELSE
C
        MJSL = 0
        CALL RHOSL(RTHP,ROP,RTHET1,ICORRL,DELRO,MJSL)
        MJ1L = MJSL
        RO1L = ROMIN + (MJ1L * DELRO) + (DELRO/2.0)
        MJSL = 0
        CALL RHOSL(RTHP,ROP,RTHET2,ICORRL,DELRO,MJSL)
        MJ2L = MJSL
        RO2L = ROMIN + (MJ2L * DELRO) + (DELRO/2.0)
        MJSH = 0
        CALL RHOSH(RTHP,ROP,RTHET1,ICORRH,DELRO,MJSH)
        MJ1H = MJSH
        RO1H = ROMIN + (MJ1H * DELRO) + (DELRO/2.0)
        MJSH = 0
        CALL RHOSH(RTHP,ROP,RTHET2,ICORRH,DELRO,MJSH)
        MJ2H = MJSH
        RO2H = ROMIN + (MJ2H * DELRO) + (DELRO/2.0)
       ENDIF
C
360    IF (HA(IBAR1,MJ1L).EQ.0) THEN
          MJ1L = MJ1L + 1
          IF (MJ1L.LE.HARO-1) GOTO 360
       ELSE
          ICNTR1 = 1
          MJ1L = MJ1L + 1
365       IF (HA(IBAR1,MJ1L).EQ.0) THEN
             ICNTR1 = 0
             MJ1L = MJ1L + 1
             GO TO 360
          ELSE
            ICNTR1 = ICNTR1 + 1
            IF (ICNTR1.NE.ICONSECL) THEN
            MJ1L = MJ1L + 1
               GO TO 365
```

```
            ELSE
             MJ1L = MJ1L - ICONSECL + 1
            ENDIF
          ENDIF
          RO1L = ROMIN + (MJ1L * DELRO) + (DELRO/2.0)
        ENDIF
370     IF (HA(IBAR1,MJ1H).EQ.0) THEN
          MJ1H = MJ1H - 1
          IF (MJ1H.GT.0) GOTO 370
        ELSE
          ICNTR2 = 1
          MJ1H = MJ1H - 1
375       IF (HA(IBAR1,MJ1H).EQ.0) THEN
            ICNTR2 = 0
            MJ1H = MJ1H - 1
            GO TO 370
          ELSE
            ICNTR2 = ICNTR2 + 1
            IF (ICNTR2.NE.ICONSECH) THEN
          MJ1H = MJ1H - 1
              GO TO 375
            ELSE
              MJ1H = MJ1H + ICONSECH - 1
            ENDIF
          ENDIF
          RO1H = ROMIN + (MJ1H * DELRO) + (DELRO/2.0)
        ENDIF
380     IF (HA(IBAR2,MJ2L).EQ.0) THEN
          MJ2L = MJ2L + 1
          IF (MJ2L.LE.HARO-1) GOTO 380
        ELSE
          ICNTR3 = 1
          MJ2L = MJ2L + 1
385       IF (HA(IBAR2,MJ2L).EQ.0) THEN
            ICNTR3 = 0
            MJ2L = MJ2L + 1
            GO TO 380
          ELSE
            ICNTR3 = ICNTR3 + 1
            IF (ICNTR3.NE.ICONSECL) THEN
          MJ2L = MJ2L + 1
              GO TO 385
            ELSE
              MJ2L = MJ2L - ICONSECL + 1
            ENDIF
          ENDIF
          RO2L = ROMIN + (MJ2L * DELRO) + (DELRO/2.0)
        ENDIF
390     IF (HA(IBAR2,MJ2H).EQ.0) THEN
          MJ2H = MJ2H - 1
          IF (MJ2H.GT.0) GOTO 390
        ELSE
```

```
            ICNTR4 = 1
            MJ2H = MJ2H - 1
 395        IF (HA(IBAR2,MJ2H).EQ.0) THEN
               ICNTR4 = 0
               MJ2H = MJ2H - 1
               GO TO 390
            ELSE
               ICNTR4 = ICNTR4 + 1
               IF (ICNTR4.NE.ICONSECH) THEN
            MJ2H = MJ2H - 1
                  GO TO 395
               ELSE
                  MJ2H = MJ2H + ICONSECH - 1
               ENDIF
            ENDIF
            RO2H = ROMIN + (MJ2H * DELRO) + (DELRO/2.0)
         ENDIF
C        WRITE(6,*) 'ICNTR1  = ', ICNTR1
C        WRITE(6,*) 'ICNTR2  = ', ICNTR2
C        WRITE(6,*) 'ICNTR3  = ', ICNTR3
C        WRITE(6,*) 'ICNTR4  = ', ICNTR4
C
 400     CO1X = ABS(RO1L * SIN2A2 - RO2L * SIN1A2)/ABS(SINDTH)
         CO1Y = ABS(RO2L * COS1A2 - RO1L * COS2A2)/ABS(SINDTH)
         CO2X = ABS(RO1H * SIN2A2 - RO2H * SIN1A2)/ABS(SINDTH)
         CO2Y = ABS(RO2H * COS1A2 - RO1H * COS2A2)/ABS(SINDTH)
         IF (THP.LT.45.0) THEN
            X1C = CO1X
            Y1C = CO1Y
            X2C = CO2X
            Y2C = CO2Y
         ELSE
            X1C = CO2X
            Y1C = CO2Y
            X2C = CO1X
            Y2C = CO1Y
         ENDIF
C        WRITE (6,*) X1C, Y1C, X2C, Y2C
C ***************************************************************
C     VERIFICATION OF COMPUTED COORDINATES OF THE ENDPOINTS
C ***************************************************************
         DXY1 = ABS(COS(RTHP)*X1C + SIN(RTHP)*Y1C -ROP)
         DXY2 = ABS(COS(RTHP)*X2C + SIN(RTHP)*Y2C -ROP)
C        IF(DXY1.GE.WIDTH) THEN
         IF(DXY1.GE.TR) THEN
C          WRITE(6,*) 'MJ1L =',MJ1L, 'MJ2L =',MJ2L
C          WRITE(6,*) 'MJ1H =',MJ1H, 'MJ2H =',MJ2H
           IF(THP.LT.45.0) THEN
            CALL LOWER(IBAR1,IBAR2,MJ1L,MJ2L,MJ1H,MJ2H,HARO,HARO1,TR,
     * HATH,HA,THP,ROP,RTHP,SIN1A2,SIN2A2,COS1A2,COS2A2,SINDTH,DELRO,
     * X1C,Y1C,X2C,Y2C)
           ELSE
```

```
          CALL HIGHER(IBAR1,IBAR2,MJ1L,MJ2L,MJ1H,MJ2H,HARO,HARO1,TR,
     * HATH,HA,THP,ROP,RTHP,SIN1A2,SIN2A2,COS1A2,COS2A2,SINDTH,DELRO,
     * X1C,Y1C,X2C,Y2C)
          ENDIF
        ELSE
C         WRITE(6,*)  'X1C = ', X1C , ' Y1C = ', Y1C
        ENDIF
C
C       IF(DXY2.GE.WIDTH) THEN
        IF(DXY2.GE.TR) THEN
C         WRITE(6,*) 'MJ1L =',MJ1L, 'MJ2L =',MJ2L
C         WRITE(6,*) 'MJ1H =',MJ1H, 'MJ2H =',MJ2H
          IF(THP.LT.45) THEN
           CALL HIGHER(IBAR1,IBAR2,MJ1L,MJ2L,MJ1H,MJ2H,HARO,HARO1,TR,
     * HATH,HA,THP,ROP,RTHP,SIN1A2,SIN2A2,COS1A2,COS2A2,SINDTH,DELRO,
     * X1C,Y1C,X2C,Y2C)
          ELSE
           CALL LOWER(IBAR1,IBAR2,MJ1L,MJ2L,MJ1H,MJ2H,HARO,HARO1,TR,
     * HATH,HA,THP,ROP,RTHP,SIN1A2,SIN2A2,COS1A2,COS2A2,SINDTH,DELRO,
     * X1C,Y1C,X2C,Y2C)
          ENDIF
        ELSE
C         WRITE(6,*)  'X2C = ', X2C , ' Y2C = ', Y2C
        ENDIF
C         WRITE(6,*)  'X1C = ', X1C , ' Y1C = ', Y1C
C         WRITE(6,*)  'X2C = ', X2C , ' Y2C = ', Y2C
C ***************************************************************
C ***************************************************************
C       COMPUTATION OF ERRORS
C ***************************************************************
C
        LENCA2 = SQRT((X1C-X2C)**2 + (Y1C-Y2C)**2)
        ERR1A2 = SQRT((X1A-X1C)**2 + (Y1A-Y1C)**2)
        ERR2A2 = SQRT((X2A-X2C)**2 + (Y2A-Y2C)**2)
C       ELENA2 = ((LENA - LENCA2)/LENA)*100.0
        ELENA2 = ABS(LENA - LENCA2)
C       ELENA2 = (LENA - LENCA2)
        ERX1 = ABS(X1A - X1C)
        ERY1 = ABS(Y1A - Y1C)
        ERX2 = ABS(X2A - X2C)
        ERY2 = ABS(Y2A - Y2C)
        IDBAR = ABS(IBAR2 - IBAR1)
        ERRL(IDBAR) = ERRL(IDBAR) + ELENA2
        ERRXY1(IDBAR) = ERRXY1(IDBAR) + ERR1A2
        ERRXY2(IDBAR) = ERRXY2(IDBAR) + ERR2A2
C ***************************************************************
        WRITE (13,670) THETA,IDBAR,ELENA2,ERX1,ERY1,ERR1A2,ERX2,ERY2,
     * ERR2A2
C ***************************************************************
C610    FORMAT(1X,F4.1,3X,I3,' , ',I3,2X, 5(2X,F7.2))
C620    FORMAT(26X, F7.3,11X,F7.3,7X,F7.3)
C ***************************************************************
```

```
 670    FORMAT(1X,F4.1,2X,I2,2X,F6.3,2X,3(1X,F5.2),2X,3(1X,F5.2))
C ************************************************************
C      WRITE (12,*) IDBAR, ELENA2
C ************************************************************
C    VERIFICATION OF ENDPOINTS DUE TO PRESENCE OF NOISE
C ************************************************************
C      RTHN = ATAN((Y2C-Y1C)/(X2C-X1C))
C      THN = ABS((RTHN/PI)*180.0)
C      THN = ABS(THN - 90.0)
C      A1 = ABS(Y2C - Y1C)
C      B1 = ABS(X1C - X2C)
C      C1 = ABS(X2C * Y1C - X1C * Y2C)
C      RON = C1/(SQRT(A1 ** 2 + B1 ** 2))
C      WRITE (6,*) 'X1A =', X1A, ' Y1A =', Y1A
C      WRITE (6,*) 'X1C =', X1C, ' Y1C =', Y1C
C      WRITE (6,*) 'X2A =', X2A, ' Y2A =', Y2A
C      WRITE (6,*) 'X2C =', X2C, ' Y2C =', Y2C
C      WRITE (6,*) 'DIFFERENCE IN BARS IN THETA DIRECTION =', IDBAR
C      WRITE (6,*) 'COMPUTED THETA IN PRESENCE OF NOISE =', THN
C      WRITE (6,*) 'COMPUTED RHO IN PRESENCE OF NOISE   =', RON
       RETURN
       END
C ************************************************************
C
       SUBROUTINE RHOSL1(RTHP,ROP,RTHD,ICORRL1,DELRO,MJSL)
       ROSL = 0.0
       XY0 = (ROP) / (COS(RTHP))
       ROSL = (COS(RTHD)) * (XY0)
       MJSL = INT( (ROSL) / (DELRO) )
       MJSL = MJSL - ICORRL1
       RETURN
       END
C
       SUBROUTINE RHOSH1(RTHP,ROP,RTHD,ICORRH1,DELRO,MJSH)
       ROSH = 0.0
       XYMAX = ((ROP) - (127.0 * SIN(RTHP))) / (COS(RTHP))
       ROSH = (XYMAX * COS(RTHD)) + (127.0 * SIN(RTHD))
       MJSH = INT( (ROSH) / (DELRO) )
       MJSH = MJSH + ICORRH1
       RETURN
       END
C
C
       SUBROUTINE RHOSL(RTHP,ROP,RTHD,ICORRL,DELRO,MJSL)
       ROSL = 0.0
       YX0 = (ROP) / (SIN(RTHP))
       ROSL = (SIN(RTHD)) * (YX0)
       MJSL = INT( (ROSL) / (DELRO) )
       MJSL = MJSL - ICORRL
       RETURN
       END
C
```

```
      SUBROUTINE RHOSH(RTHP,ROP,RTHD,ICORRH,DELRO,MJSH)
      ROSH = 0.0
      YXMAX = ((ROP) - (127.0 * COS(RTHP))) / (SIN(RTHP))
      ROSH = (127.0 * COS(RTHD)) + (YXMAX * SIN(RTHD))
      MJSH = INT( (ROSH) / (DELRO) )
      MJSH = MJSH + ICORRH
      RETURN
      END
C
      SUBROUTINE LOWER(IBAR1,IBAR2,MJ1L,MJ2L,MJ1H,MJ2H,HARO,HARO1,TR,
     * HATH,HA,THP,ROP,RTHP,SIN1A2,SIN2A2,COS1A2,COS2A2,SINDTH,DELRO,
     * X1C,Y1C,X2C,Y2C)
      INTEGER HARO, HATH, HARO1
      INTEGER*2 HA(0:HATH-1,0:HARO1-1)
      ROMIN = 0.0
      DO 800 L2 = MJ2L,MJ2H
         DO 750 L1 = MJ1L,MJ1H
            RO1L = ROMIN + (L1 * DELRO) + DELRO/2.0
            RO2L = ROMIN + (L2 * DELRO) + DELRO/2.0
            C2X1 = ABS(RO1L * SIN2A2 - RO2L * SIN1A2)/ABS(SINDTH)
            C2Y1 = ABS(RO2L * COS1A2 - RO1L * COS2A2)/ABS(SINDTH)
            DXY12 = ABS(COS(RTHP) * C2X1 + SIN(RTHP) * C2Y1 - ROP
            IF(DXY12.LT.TR) GO TO 820
750      CONTINUE
800   CONTINUE
      WRITE(6,*) 'ENDPOINT CANNOT BE COMPUTED'
C     GO TO 825
820   IF(THP.LT.45.0) THEN
        X1C =  C2X1
       Y1C =  C2Y1
      ELSE
        X2C =  C2X1
       Y2C =  C2Y1
      ENDIF
825   RETURN
      END
C
      SUBROUTINE HIGHER(IBAR1,IBAR2,MJ1L,MJ2L,MJ1H,MJ2H,HARO,HARO1,TR,
     * HATH,HA,THP,ROP,RTHP,SIN1A2,SIN2A2,COS1A2,COS2A2,SINDTH,DELRO,
     * X1C,Y1C,X2C,Y2C)
      INTEGER HARO, HATH, HARO1
      INTEGER*2 HA(0:HATH-1,0:HARO1-1)
      ROMIN = 0.0
      DO 900 LH2 = MJ2H,MJ2L,-1
         DO 850 LH1 = MJ1H,MJ1L,-1
            RO1H = ROMIN + (LH1 * DELRO) + DELRO/2.0
            RO2H = ROMIN + (LH2 * DELRO) + DELRO/2.0
            C2X2 = ABS(RO1H * SIN2A2 - RO2H * SIN1A2)/ABS(SINDTH)
            C2Y2 = ABS(RO2H * COS1A2 - RO1H * COS2A2)/ABS(SINDTH)
            DXY22 = ABS(COS(RTHP) * C2X2 + SIN(RTHP) * C2Y2 - ROP
            IF(DXY22.LT.TR) GO TO 920
850      CONTINUE
```

```
900     CONTINUE
        WRITE(6,*) 'ENDPOINT CANNOT BE COMPUTED'
C       GO TO 925
920     IF(THP.LT.45.0) THEN
          X2C =  C2X2
         Y2C =  C2Y2
        ELSE
          X1C =  C2X2
         Y1C =  C2Y2
        ENDIF
925     RETURN
        END
```