SKEW DETECTION USING HOUGH TRANSFORM

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Computer Science and Engineering

Thapar University, Patiala

By:
Arun Arora
(80632003)

Under the supervision of:
Mr. Ravinder Kumar
Lecturer
CSED, Thapar University

MAY 2008

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, **"Skew Detection using Hough Transform"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in Computer Science and Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Mr. Ravinder Kumar* and refers other researcher's works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

(***Arun Arora)***

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(**Mr. Ravinder Kumar)**
Computer Science and Engineering Department
Thapar University
Patiala

Countersigned by

(Dr. (Mrs.) Seema Bawa)
Professor & Head
Computer Science & Engineering. Department
Thapar University
Patiala

(Dr. R.K.Sharma)
Dean(Academic Affaris)
Thapar University,
Patiala.

ACKNOWLEDGEMENT

"The need to be right all the time is the biggest bar to new ideas. It is better to have enough ideas for some of them to be wrong than to be always right by having no ideas at all."

- Edward de Bono

I am highly thankful to my guide Mr. Ravinder Kumar, Lecturer, Computer Science & Engineering Department, Thapar University, Patiala, for his advice, motivation, guidance, moral support, efforts and the attitude with which he solved all of my queries in making this thesis possible. It has been a great honor to work under him.

I am also thankful to Dr. (Mrs.) Seema Bawa, Professor and Head**,** Computer Science and Engineering Department, Thapar University, Patiala, for providing us with adequate infrastructure in carrying the research work.

I am also thankful to Mrs. Damandeep Kaur (PG Coordinator) and other faculty members for their contribution in making this thesis possible. I am also grateful to Mr. Nitin Bhatia who helps me in understanding the mathematical foundations needed for this thesis work. Besides the above dignitaries, I am also thankful to my friend Sham and Ashish Jain for their kind and noble support.

I am forever indebted to my parents for their constant support and encouragement throughout the past years. This thesis is devoted to them.

Arun Arora

80632003

ABSTRACT

Document image processing has become an increasingly important technology in the automation of office documentation tasks. Automatic document scanners such as text readers and OCR (Optical Character Recognition) systems are an essential component of systems capable of those tasks. One of the problems in this field is that the document to be read is not always placed correctly on a flatbed scanner. This means that the document may be skewed on the scanner bed, resulting in a skewed image. This skew has a detrimental effect on document analysis, document understanding, and character segmentation and recognition. Consequently, detecting the skew of a document image and correcting it are important issues in realizing a practical document reader.

Very frequently the digitalization process of documents produce images rotated of small angles in relation to the original image axis. The skew introduced makes more difficult the visualization of images by human users. Besides that, it increases the complexity of any sort of automatic image recognition, degrades the performance of OCR tools, increases the space needed for image storage, etc. Thus, skew correction is an important part of any document processing system being a matter of concern of researchers for almost two decades now. The search for faster and good quality solutions to this problem is still on.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE **Page No.**

LIST OF TABLES

TABLE **Page No.**

CHAPTER 1
INTRODUCTION

1.1 Background

Organizations are moving at a fast pace from paper to electronic documents. However, large amounts of paper documents inherited from a recent past are still needed. Digitalization of documents appears as a bridge over the gap of past and present technologies. Scanners tend to be of widespread use for the digitalization of documents. One of the important problems in this field is that very often documents are not always correctly placed on the flat-bet scanner either manually by operators or by the automatic feeding device. This very frequent problem yields rotated images. For humans, rotated images are unpleasant for visualization and introduce extra difficulty in text reading. For machine processing, image skew brings a number of problems that range from needing extra space for storage to making more error prone the recognition and transcription of the image by automatic OCR tools. These reasons make skew detection and correction phases a common place in any environment for document processing.

Very frequently the digitalization process of documents produce images rotated of small angles in relation to the original image axis. The skew introduced makes more difficult the visualization of images by human users. Besides that, it increases the complexity of any sort of automatic image recognition, degrades the performance of OCR tools, increases the space needed for image storage, etc. Thus, skew correction is an important part of any document processing system being a matter of concern of researchers for almost two decades now. The search for faster and good quality solutions to this problem is still on.

1. 2 Document Image Processing Steps

In document analysis the first step is to acquire a digitized raster image of the document using a suitable scanning system. Then it is followed by page layout analysis and character recognition. Before the structure of the text is obtained, a test is carried out to

find out whether the document is skewed. Then skew is corrected and thereafter character recognition is done.

1. **Scanning The Document**
2. **Skew Detection**
3. **Skew Removal**
4. **Page Layout Analysis And Character Recognition**

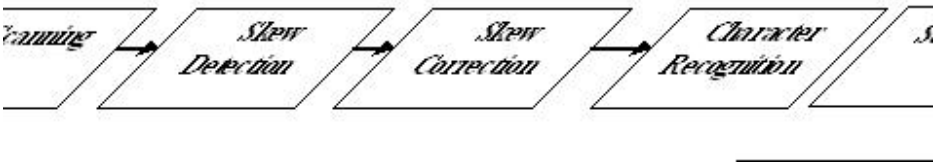


Figure 4.1: Document Image Processing Steps

Scanning: it is process of converting the paper document into digitized format.

Skew Detection: Scanned document may have skew problems. It may be skewed. So before document can be properly analyzed for character recognition, its skew ness has to be detected. The angle of skew is obtained by applying different algorithms.

Skew Removal: After the skew angle has been measured. Then next phase is to remove the detection. It is done by rotating the digitized image by skewed angle. This is done simply by rotation. If document is skewed in clockwise, then document has to be rotated in anti-clock wise direction and if document is skewed in anti-clockwise, then document has to be rotated in clock wise direction.

Character recognition: Now document is ready for character recognition

1.3 Document Skew and Its Reasons

The conversion of paper documents to electronic format is routinely done for record management, automated document delivery, document archiving, journal distribution etc.

The stages of document conversion include scanning, displaying, image processing, text recognition, image and text database creation and quality assurance. During the scanning process, the whole document or a portion of it is fed through a loose-leaf page scanner. Some times pages are not fed properly into the scanner causing skew-ness of these bitmapped-image pages. A significant skew in document can be detected by human vision easily and the skew correction can be made by re-scanning the document, whereas for mild skew it may not be possible to notice its skew as human vision system fails to identify it. Even a smallest skew angle existing in a given document image results in the failure of segmentation of complete characters from words or a text lines, as the distance between the character reduces. Further most of the OCRs and document retrieval/display systems are very sensitive to skew in document images. Following figures shows skewed scanned document.

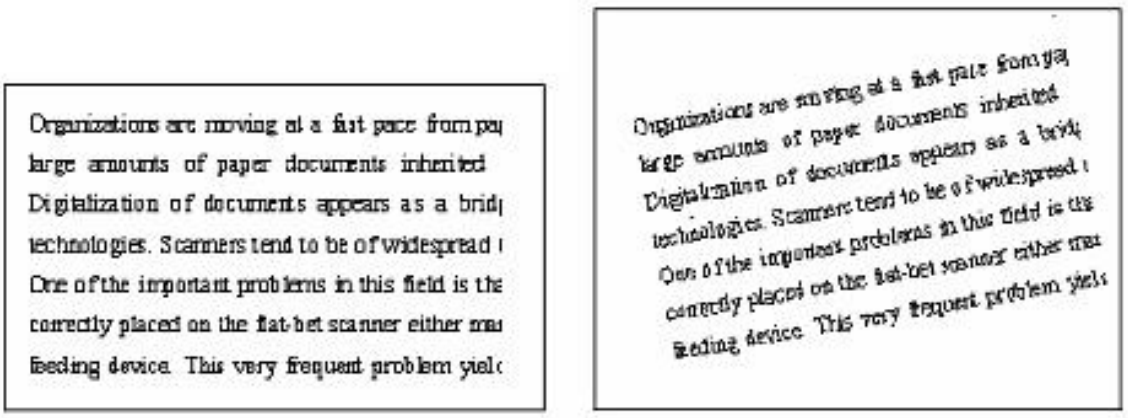
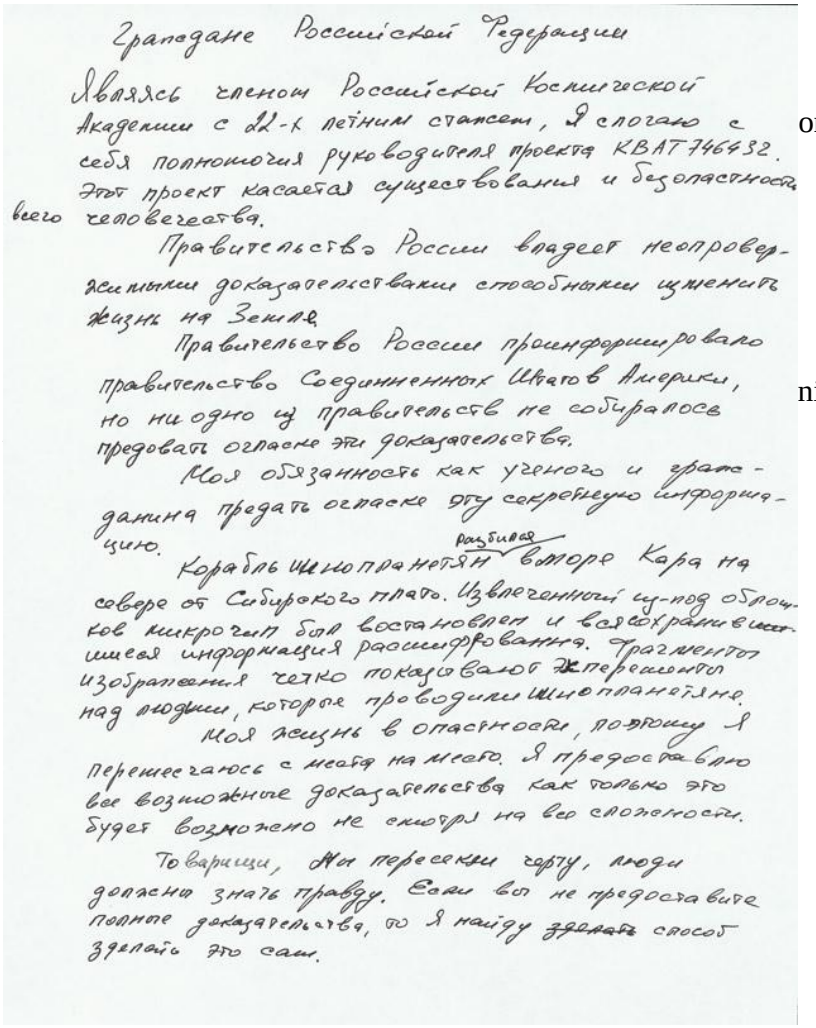


Figure 1.5: (a) original document and (b) skewed scanned Document

It is important to detect and correct skew ness. There can be many reasons for skew-ness in document images. But there are two most basic reasons are enumerated below. Skew in Scanning Process

1. **Skew in Scanning Process**
2. **Skewed Hand Writing**
3. **Skewed Original Document**



Граждане Российской Федерации

Являясь членом Российской Космической Академии с 22-х летним стажем, я слагаю с себя полномочия руководителя проекта КВАТ 746432. Этот проект касается существования и безопасности всего человечества.

Правительства России владеет неопровержимыми доказательствами способными изменить жизнь на Земле.

Правительство России проинформировало правительство Соединенных Штатов Америки, но ни одно из правительств не собиралось предавать огласке эти доказательства.

Моя обязанность как ученого и гражданина предать огласке эту секретную информацию.

Корабль Инопланетян разбился вторгся Кара на севере об Сибирского плато. Извлеченной из-под обломков микрочип был восстановлен и всех сохранившейся информации расшифрования. Фрагменты изображения четко показывают эксперименты над людьми, которая проводили Инопланетяне.

Моя жизнь в опасности, поэтому я перемещаюсь с места на место. Я предоставлю все возможные доказательства как только это будет возможно не смотря на все сложности.

Товарищи, мы пересекли черту, люди должны знать правду. Если вы не предоставите полные доказательства, то я найду способ сделаю это сам.

on for skew-ness in nition process of hand

Figure 1.3 Skewed Hand Writing

Skewed Original Document: one of the basic reasons of skew ness can be that original source of scanning document may be skewed. This can happen when we are using type writer written document or print outs taken from printer where paper has not been placed properly.

1.4 Types of Document Skew

There are two type of skew in document images

1. **Single Skew**
2. **Multiple Skew**

Single Skew: In this skew, whole document is skewed to single angle. Most of document images have this type of skew-ness. This work deals with Single Skew problem. Lot of work has been done in this field and lot of research is still going on.

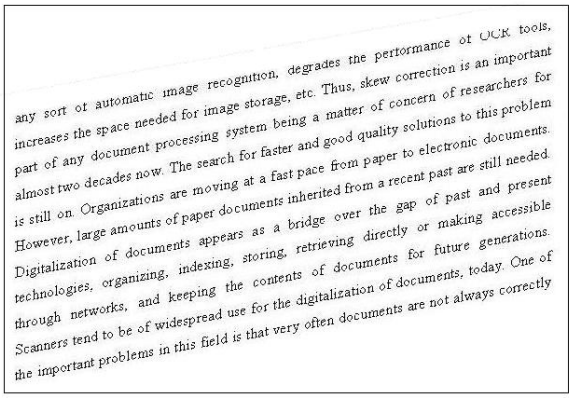


Figure 1.4: Single Skew

Multiple Skew: In this, scanned document can have many sections; each may be skewed to different angle. Detecting such type of skew-ness needs lot of efforts. Multiple Skew problem exists rarely and has not got lot attention from researchers.

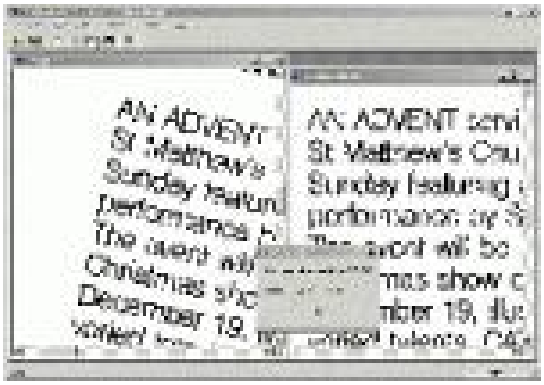


Figure 1.5: Multiple Skew

1.5 Skew Detection Approaches

Several approaches have been proposed as alternatives for skew angle detection of document images. All of them require a dominant text area to be present in order to work properly. Main approaches for skew detection include:

- **Hough transform**
- **Projection Profile**
- **Nearest neighbor**
- **Principal Component Analysis**

Hough transform is a popular method for skew detection. It is capable of locating fragmented lines in a binary image. Therefore given a group of black pixels, one can find the imaginary line or lines that go through the maximum number of these pixels. Given a binary image with a dominant text area, the detected lines will most probably go along the whole middle zone of the textual lines. Hence these lines have approximately the same skew as the reference lines of the text which the skew of the whole page. Whenever the Hough transform is used, there is always a tradeoff between accuracy and speed. The more accurate the angles of the detected lines are, the more computation is required. In addition the computation time depends on the number of pixels in the image. Connected component analysis is required. Then several iterations take place in order to connect each component to its nearest neighbor in recursion. This process results in several chains representing the textual lines. A line that goes through the central mass of the components in a chain approximates the skew of the associated textual line. Alternatively, one can calculate the skew by averaging the skews of the lines that connect neighboring components in a chain. This method is based on the fact that inter character and inter word spaces between two consecutive characters or words respectively, are usually smaller than spaces between such neighboring elements that belong to different lines.

1.5.1 Hough Transform

The Hough transform is a technique which can be used to isolate features of a particular shape within an image. Because it requires that the desired features be specified in some parametric form, the classical Hough transform is most commonly used for the detection of regular curves such as lines, circles, ellipses, etc The **Hough transform** (pronounced h?**f**/, rhymes with tough) is a feature extraction technique used in image analysis,

computer vision, and digital image processing. The purpose of the technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure. This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform.

The classical Hough transform was concerned with the identification of lines in the image, but later the Hough transform has been extended to identifying positions of arbitrary shapes, most commonly circles or ellipses. The Hough transform as it is universally used today was invented by Richard Duda and Peter Hart in 1972, who called it a "generalized Hough transform". The simplest case of Hough transform is the linear transform for detecting straight lines. In the image space, the straight line can be described as y = mx + b and can be graphically plotted for each pair of image points (x,y). In the Hough transform, a main idea is to consider the characteristics of the straight line not as image points x or y, but in terms of its parameters, here the slope parameter m and the intercept parameter b. Based on that fact, the straight line y = mx + b can be represented as a point (b, m) in the parameter space. However, one faces the problem that vertical lines give rise to unbounded values of the parameters m and b[2]. For computational reasons, it is therefore better to parameterize the lines in the Hough transform with two other parameters, commonly referred to as **?Ò**(rho) and **?Ò**(*theta*). The parameter **?•**represents the distance between the line and the origin, while **?•**is the angle of the vector from the origin to this closest point (see Coordinates).

1.5.1.1 Basic Idea

The polar representation for the line

?²= x cos ?²+ y sin ?², for 0 =?²<Π .....(1)

where **?%**is perpendicular distance of the line from the origin and,

and **?¤** angle from the horizontal of the perpendicular line.

There are two line parameters, distance (**?'),** and angle (**?ª**) represented as **(?ª, ?')** that defines the hough space. Each Points in the **(*x, y*)** domain is mapped on to curves in the **(?¡ ?)** domain as shown in figure2.

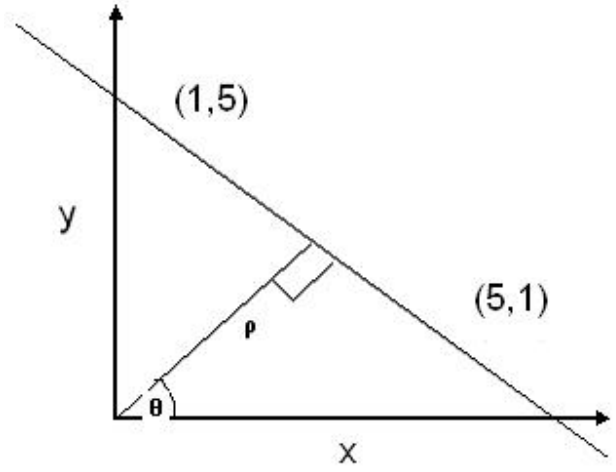


Figure 1.6: X, Y plane for line detection

Suppose we are given two points (1, 5) and (5,1) and we want to calculate angle made by the line passing through these points. Consider the first point (1, 5), we have

?® cos ?® 5* sin ?®for 0 =®Π

We will draw curve for different value of **?'**, we will get ssinusoidal curve show n fig3 (a).

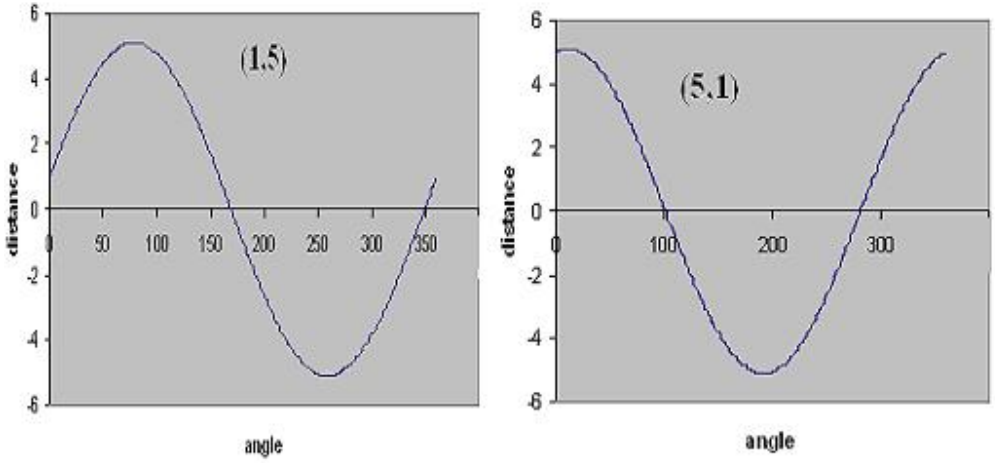


Figure 1.7: (a) sinusoidal curve of (1,5) **(b)** sinusoidal curve of (5,1)

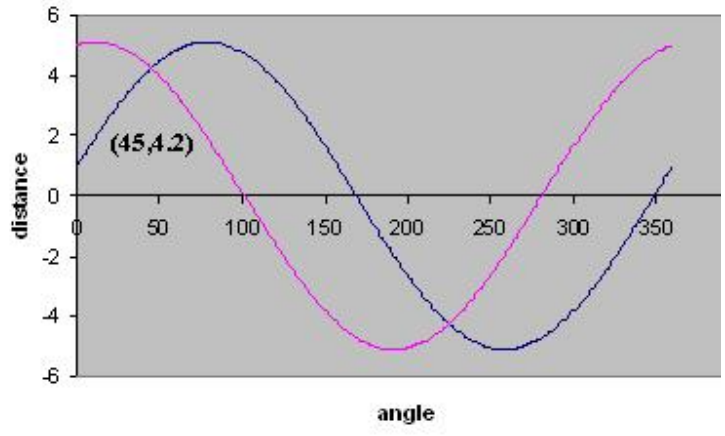


Figure 1.7: (c) Intersection of two sinusoidal curves at point ((?W?)W(45,4.24)

Now consider the second point (5, 1), we have

?í= 5*cos ?í+ sin ?¡ for 0 =?í<Π

Fig 3(b) represents sinusoidal curve for point (5, 1) and fig 3(c) shows the intersection of these two curves. This intersection shows that value of **?**"is 45^0. From this value **?;**"we can calculate angle made by this line on X-axis.

1.5.1.2 Cost of Hough Transform

Time complexity: The running cost is **O (n×A**), where n is number of points and **A** is number of different values of angles. So more accuracy we need, then more fine angle intervals we have to use and hence more different values for angle, and more the running time

Space Complexity: Accuracy Hough Transform heavily depends upon how much small is step size for distance parameter **?¢**suppose **?¢**can take values up to10 and we divide it in ten intervals step size of one. Then there can be 11 values for **?** from 0, 1,….up to 10 and space requirement for Hough Matrix **H** will be **A ×11** and if we take step size of 0.5 for **? ,** then there can be 21 different values for it and space requirement for Hough Matrix **H** will be **A ×21.** Hence space requirement depends upon accuracy desired.

Smallest the step size for **?ª¸** more the accuracy will and more the space for Hough Matrix is required.

1.5.1.3 Properties of Hough Transform

(1) Hough Transform is "Voting" algorithm.

(2) Since each point is handled independently, parallel implementations are possible.

(3) Hough is computationally expensive algorithm and so it is advised to preprocess the document to reduce the number of pixels in order to reduce the processing. So current research is going on to find the most appropriate pixels form document image to be used for determining orientation of document using Hough Transform

1.5.1.4 Reducing Number of pixels

The running cost is **O (n×A**), where n is number of points and **A** is number of different values of angles. So more accuracy we need, then more fine angle intervals we have to use and hence more different values for angle, and more the running time. As complexity is directly proportional to number of foreground pixels, so effort is done to reduce the number of input pixels. So Hough transform is rarely performed directly on input image. So input image is preprocessed before applying Hough Transform. Following fig shows how this works. It is important how we filter pixels from image document. There have been many algorithms that uses Hough Transform for skew detection. They mainly differ in aspect that how they do preprocessing before applying Hough Transform. For example Amin [19] uses connected component analysis followed by grouping of connected components. Connected components are bounded by rectangular boundary representing connected black pixels Then Hough Transform is applied to resulting document image. M. Aradhya [21] also discusses the skew detection technique based on Hough Transform. It uses Thinning as preprocess step. Thinning is process of reducing thickness of each line of pattern to just single pixel. This algorithm is iterative and it deletes every point on that lies on outer boundary of symbol.

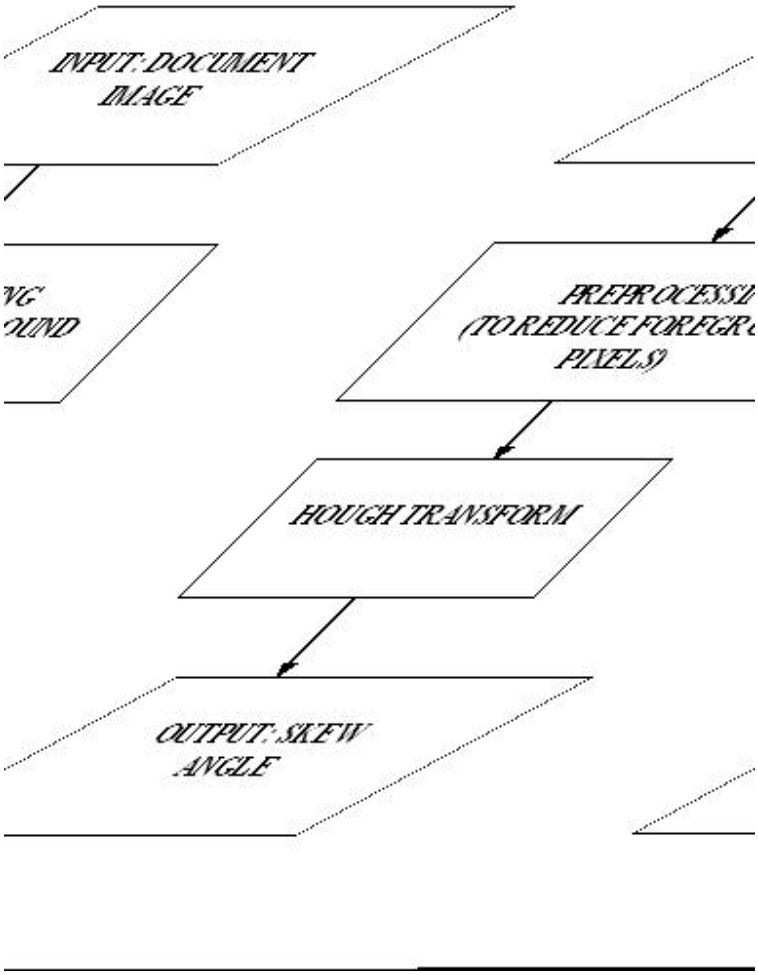


Figure 1.8 Basics Steps of Algorithms that uses Hough Transform

1.5.2 Projection Profile

The traditional projection profile approach was proposed by Postl [4]. In this approach, the input document is rotated through a range of angles and a

Projection profile is calculated at each angle [2] Features are then extracted from each projection profile to determine the skew angle. As we know that baseline is the part of the script having most of the black pixels. Projection profile based technique [3-8] takes the advantage of this property to find the space between the lines of the script. Then the tilt

angle using projection and rotation can easily be estimated. Projection Profile based algorithm works as follows [2]:

1. Start with (?ê=0) and project horizontally and vertically to get the sum of the black pixels in the binary image and record the max of these two sums as well as ?Z
2. Increase ?¹by 1 and rotate the image clockwise and project to get the maximum of the two sums. If the current sum is greater than the previous one then update the maximum sum and ?vother wise go to the next ?v
3. Repeat Step 2 till ?Ë90.
4. Rotate the image either by ?4or by ?4-90 depending on whether the maximum are gotten horizontally or vertically respectively.

In projection profiles, where a histogram is created at each possible angle and a 'cost function' is applied to this histogram. The skew angle is the angle at which this cost function is maximized.

1.5.3 Nearest Neighbor

Hashizume et al [9] propose a method that computes the nearest neighbor of each character and creates a histogram of these values to detect the skew angle. . He found all the connected components in the document and computed the direction of its nearest neighbor for each component. A histogram of the direction angle is computed, the peak of which indicates the document skew angle. Liu et al [10] use a similar technique to Hashizume et al, in which they detect and remove the characters with ascenders or descenders and then calculate the inclination between adjacent base points of connected components. These two methods rely on the information that in-line spacing is less than between-line spacing.

1.5.4 Principal Component Analysis

Principal component analysis (PCA) [11] is a technique used to reduce multidimensional data sets to lower dimensions for analysis. Depending on the field of application, it is also

named the **Hotelling transform** or **proper orthogonal decomposition (POD)**.

PCA was invented in 1901 by Karl Pearson. Now it is mostly used as a tool in exploratory data analysis and for making predictive models. PCA involves the calculation of the eigenvalue decomposition of a data covariance matrix or singular value decomposition of a data matrix, usually after mean centering the data for each attribute. PCA is mathematically defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on. PCA is theoretically the optimum transform for a given data in least square terms. PCA can be used for dimensionality reduction in a data set by retaining those characteristics of the data set that contribute most to its variance, by keeping lower-order principal components and ignoring higher-order ones. Such low-order components often contain the "most important" aspects of the data. However, depending on the application this may not always be the case

Principal Components Analysis (PCA) is a way of finding the directions along which a distribution exhibits the greatest variation [12]. These directions are termed in the Principal Components of the distribution. They correspond to the most significant eigenvectors of the covariance matrix of the data points. PCA is used to find the principal axis of the foreground distribution. This gives an angle for the baseline but not necessarily its vertical position. Steinherz [13] used this method on Latin text. It was found that the performance is high when using the background pixels for PCA rather than the foreground. This method [2] works as follows: First of all Convert the image into a set of vectors describing each foreground pixel that makes up the text. Then PCA is performed on these points and eigenvector with the largest eigen value is chosen. This vector gives the direction of the baseline. Image is rotated so that this estimated baseline angle lies horizontal. Projection histogram of the image is taken then and peak of this histogram is used to determine the vertical position of the baseline.

CHAPTER 2
LITERATURE REVIEW

Digitalization of paper documents has become very important in field of office automation systems. Organizations are moving from printed paper to digitized documents at very fast pace. Digitalization of existing paper documents goes through scanning, skew detection, skew removal and then recognition. Optical Character Recognition (OCR) devices are widely used for this recognition purpose. Most often, skew is introduced during scanning due to improper placement of paper on scanner or due to original rotated documents. This skew has a detrimental effect on document analysis, document understanding, and character recognition. Consequently, detecting the skew of a document image and correcting it are important issues in realizing a practical document reader. A variety of algorithms have been devised to tackle this problem.

Skew Detection has become very important as part of document processing tools. Skew has detrimental impact on document analysis. This problem has been addressed by many researchers for more than two decades and several approaches have been devised. These approaches can be categorized as Hough Transform (HT), Correlation, Nearest Neighbor, projection Profile etc. Hough Transform has been widely used technique. Hough Transform is computationally expensive algorithm. In addition the computation time depends on the number of pixels in the image and number of different value of angle. Therefore many methods ([14], [15], [16]) make some efforts to reduce the number of pixels by compressing a group of pixels into one representative. Yu [17] uses hierarchical Hough Transform. Basic purpose is to reduce the input data. Hough Transform provides accuracy but on the cost of heavy computational time. Preprocessing can be done to reduce such image points by using some criteria. Amin[19] uses connected component analysis followed by grouping algorithm[20] to reduce the input pixels. M. Aradhya [21] also discusses the skew detection technique based on Hough Transform. It also finds the connected components from the document image, and filter out the small components. Selected components are blocked and thinning operation is performed on selected blocks hence it becomes important that how we select the points for the Hough transform. So

this preprocessing or filtering has biggest impact on accuracy of algorithm. Yan [18] proposed the cross-correlation method. In two parallel lines, pixels are translated due to skew, and then correlation matrix is produced. In following section I have discussed some important skew detection algorithms. First two algorithms are based on Hough Transform. Third Algorithm uses Principal Component Analysis (PCA). Fourth algorithm uses projection Profile and used to determine skew of documents containing Gurmukhi Script.

2.1 A. Amin and S. Fischer's Algorithm

Amin [19] discusses method for skew determination which uses Hough Transformation. As complexity of Hough transformation is directly dependent on the no of image points. So it does some preprocessing to reduce no. of points to be used in Hough Transformation. This algorithm determine the skew angle of the entire document by identifying blocks of text, such as paragraphs or captions for pictures, and calculating a skew angle for each block. The skew angle for the entire page is then determined by choosing the most frequent angle, after applying a weighting factor to each angle to account for the number of points used in its calculation. It uses first Hough Transform to estimate the angle, and then it uses Least Squares formula to find more precise figure to determine the slope of the line that best fits the points used. Since the Hough Transform is computationally expensive, we reduce the amount of data to be used in the calculation by grouping the pixels into connected components which will generally represent characters, although noise may break a character into two or more pieces or join a group of characters into one connected component. Completely correct segmentation of characters is not vital to the algorithm, and indeed, experiments have shown that there is a wide margin for error in the segmentation process where the skew results are unaffected. Basic Steps in this Algorithm are

1. Preprocessing
2. Connected component analysis
3. Grouping
4. Skew Estimation

5. Skew Calculation
6. Skew Correction

Before any skew angle can be calculated, the image must be scanned. The test pages were stored as grey-level images so that data loss due to scanning was minimized. In connected component analysis, connected components are bounded by rectangular boundary representing connected black pixels. This is iterative procedure to form rectangles around different components, whether images or characters. It compares successive scanlines of an image to determine whether black pixels in any pair of scanlines are connected together. Bounding rectangles are extended to enclose any groupings of connected black pixels between successive scanlines.

After the determination of all connected components, connected components are grouped according to dimensions. They are categorized into three categories: noise, small and large. Noise components are ignored and connected components are merged based on prefixed threshold using grouping algorithm [20].] takes one CC at a time, and tries to merge it into a group from a set of existing groups. If it succeeds the group's dimensions are altered so as to include the new CC that is the group encompassing the rectangle is expanded so as to accommodate the new CC along with the existing CCs already forming the group. A possible merge is found when the CC and a group (both of the same category) are in close proximity to each other. When a CC is found to be near a group, its distance from each CC in the group is then checked until one is found that is within the predefined threshold distance or all CCs have been checked. If such a CC is found, then the CC is defined to be in the neighborhood of the group and is merged with that group. If the CC can not be merged with any of the existing groups then a new group is formed with its sole member being the CC which caused its creation. Figure 1 demonstrates the conditions necessary for a CC to be merged into an existing group. The small CC (hashed) is merged into the right group and the group dimensions increase to accommodate the after grouping, skew estimation is done. For this group is divided vertically into segments of width approximately equal to width of one component and store one bottom rectangle in each segment. Hough Transformation is applied to center of rectangle. We get the line of connected components that represents bottom row. Least square method is applied to points to find the slope.

$y = a + bx$

where the coefficients *a* and *b* are computed using the following formulae [22]

$$b = \frac{(n?\$xy - ?\$x?\$y)}{?\acute{a}x^2 - (?\fx)^2}$$

$$a = \frac{(?Ay - b?Ax)}{n}$$

and orientation is given by

$$\text{Angle(a)} = \tan^{-1}(b)$$

Where b represents the slope of line.

Once we obtained the angle, we can de-skew the document by rotating the document. This method was inspired by Paeth [23]. To calculate the correct value for a pixel at location **(*x*,*y*)** in the skew corrected image,

$$x_{old} = x \cos(a) \quad y\sin(a)$$

$$y_{old} = y\cos(a) \quad x\sin(a)$$

Where **a** is the calculated skew angle of the image.

Amin's Algorithm is Hough based and slower because connected components must be grouped by size. This step takes lot of time and computations. But this step improves accuracy. Hence we are getting here accuracy on the expense of time.

2.2 Manjunath Aradhya V N, Hemantha Kumar G, and Shivakumara P's Algorithm

M. Aradhya [21] also discusses the skew detection technique based on Hough Transform. It uses Thinning as preprocess step. It has two stages and then Hough Trans

1. Selected characters from document image are blocked.
2. perform thinning operation on selected block
3. Hough Transform is performed

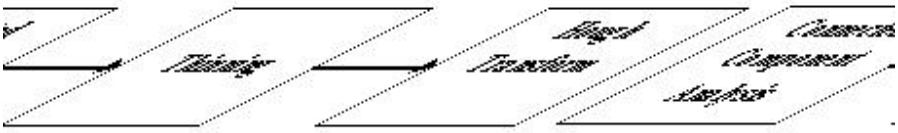


Figure 2.6 Steps of Aradhya Algorithm

It also finds the connected components from the document image, and filter out the small components. Detailed Algorithm is as follow. Connected components in the document image are found and average bounding height is computed. It uses the parameter average bounding height (AH) for filtering process. If component height is less than AH, then it is removed. Selected components are blocked and thinning operation is performed on selected blocks and straight lines present in the document are removed using predetermined threshold. Then resultant points are passed to Hough Transform. From Hough transform, we get the orientation of document image. However this method fails if the noise density increases by 0.05. One big disadvantage of this algorithm is that it fails on document that contains images. Maher and Ward Thinning [24] algorithm is used here. Thinning is process of reducing thickness of each line of pattern to just single pixel. This algorithm is iterative and it deletes every point on that lies on outer boundary of symbol. This algorithm defines 20 rules for thinning process.

2.3 Steinherz T, Intrator N and Rivlin E's Algorithm

This Method[25] uses Principal Component Analysis. This method also finds the exact angle at which projection profile has maximum deviation, within single iteration. Given a binary image, where stroke pixels are black and background pixels are white, one can map any black pixel in the image to a two dimensional vector and vice versa. For example, a black pixel located in raw y and column x in the image is mapped to the

vector (x; y). The projection profile of the image on any axis, is equal to the projection profile of the set of associated vectors on a unit vector that points to the same direction as the axis does. In this case one can find the unit vector that maximizes the projection profile deviation as an equivalent skew detection method. The desired unit vector is called the Principal Component of the given vector set, and it is equal to the eigen vector that is associated with the largest absolute eigen value of this set. This property is important because it implies that the skew angle can be found within one iteration Therefore the proposed algorithm is the following. Given binary document image, we create an isomorphic set of two dimensional vectors, where each vector corresponds to a black pixel in the image. The transformation maps a pixel to a vector of the same coordinates. Then the Principal Component of the resulted vector set is found: first the eigen values and corresponding eigen vectors of the vector set are found, then the vector associated with the largest absolute eigen value is picked. The direction of the Principal Component found is perpendicular to the direction of the text lines in the image, which is referred to as the skew angle.

2.4 Skew Detection Algorithm for Gurmukhi Script by G S Lehal and Renu Dhir

Lehal and Dhir [26] proposed algorithm for detection of skew in documents containing Gurmukhi script. This algorithm is based on projection profile. Gurmukhi script is used primarily for the Punjabi language. It is spoken by 84 million native speakers and is the world's 14th most widely spoken language. The Gurmukhi script alphabet consists of 40 consonants and 12 vowels writing style is from left to right. The concept of upper/lower case is absent. Gurmukhi script can be partitioned into two horizontal zones. The upper zone denotes the region above the head line, while the lower zone represents the area below the head line (Fig 2). The major part of the characters is located in the lower zone.

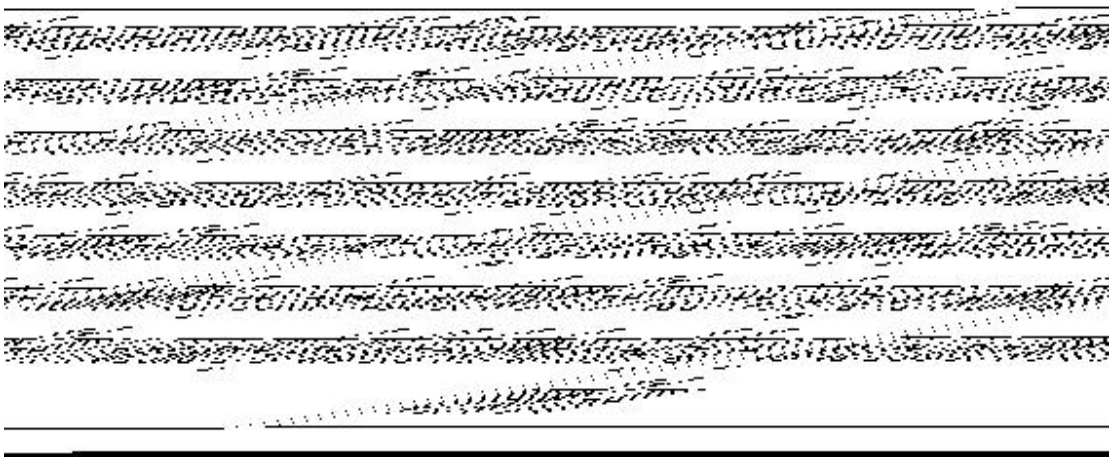


Figure 7.2 Gurmukhi script

Above fig shows an example of Gurmukhi script. The skew angle is determined by calculating horizontal and vertical projections at different angles at fixed interval in range [0°, 90°]. A horizontal projection maps a two-dimensional bitmap into one-dimensional by recording the number of foreground pixels in each row of the bitmap in a one dimensional array, where an entry in the array corresponds to the y coordinate of the row in the bitmap [26]. Similarly the vertical projection keeps track of foreground pixel in each column. Under such projections, for an image with no skew, headlines appear as distinct peaks while gaps between successive text rows will be represented by valleys our task is to determine the angle at which the highest peaks and deepest valleys in the projections are present. Both the horizontal and vertical profiles are simultaneously examined for peaks and valleys. According to empirical study, 10 peaks and 10 valleys at each angle are identified, assuming there are at least 10 lines of text, and a measure of difference of sum of heights of peaks and valleys was made and the angle at which the difference was maximum was recorded. It was observed that if the document contained large strips of rows or columns of foreground and background pixels, then the highest peaks and valleys occurred in consecutive rows or columns and thus highest peaks and valleys were detected in skewed image also, resulting in wrong skew angle estimation. To overcome this problem, the bitmapped image is partitioned into ten equal sized horizontal and vertical zones and highest peaks and valleys determined for projections in each zone. The angle, at which the difference of the sum of heights of peaks and valleys is maximum, is identified as the skew angle. One major advantage of this algorithm is

that any document with orientation portrait or landscape can automatically be made straight.

2.5 Baird's Algorithms

Baird's algorithm [27] detects the skew angle by using the projection profile technique. The method is said to work on a wide variety of layouts, including multiple columns, sparse Baird claims that his method is one of the fastest and most accurate skew detection algorithms. First, Baird applies a connected component analysis to the document. The midpoint of the bottom of each connected component is then projected onto an imaginary accumulator line perpendicular to different projection angles. Large connected components like images will be ignored during the projection process; while the other connected components such as the characters, character fragments and margin noise are projected onto the accumulator line.

2.6 Hinds et al Algorithm

Hinds et al [28] apply the vertical runlength analysis for the image. A grey-scale 'burst image' is created from the black runlengths that are perpendicular to the text lines by placing

the length of the run in the run's bottom-most pixel. The Hough Transform is then applied to each of these burst images. Runlengths with values between 1 and 25 are being applied with the Hough Transform. Unlike the standard approach of incrementing the accumulator cells by one, the cells are incremented by the value of the burst image.

Since the Hough Transform is computationally expensive and is slowed by noise, this algorithm reduces the amount of data within a document image through the computation of its vertical black runlengths. This algorithm de-emphasises the noise and emphasises the contribution of the bottoms of the lines of text. This method is said to work up to 45°. The accuracy depends upon the angular resolution of the Hough transform.

2.7 O'Gorman's Algorithm

Gorman[29] discusses his skew detection method as part of a structural page layout analysis system, which detects the in-line and between-line spacings, and locates text lines and blocks as well as calculating the skew angle. This means that computing time for determining skew is not necessarily optimum. The method involves a bottom-up nearest neighbor clustering.

2.8 Postl's Algorithms

Postl [30] discusses two methods for skew determination. Postl calls his first method the 'simulated skew scan' method. It amounts to maximizing a chosen cost function as the slope of the projection profile is varied. The cost function is the total variation of the projection profile, i.e. the sum of the squares of the successive differences of the projection profile bin heights. Postl's second method involves taking a two-dimensional Fourier transform of the pixel density distribution. In practice, this is a discrete transformation using the chosen bin widths. The projection profile can be expressed as the (single) integral along the radius vector of the Fourier transform with an imaginary exponential kernel. Hence, the two concepts are directly related and any suitable cost function can be employed for the optimization and determination of the skew angle. The derivative of the projection profile corresponds to the successive differences between bins in the discrete analogue of Postl. The power spectrum is given by the magnitude of the transformed function Z squared.

CHAPTER 3
PROBLEM STATEMENT

Hough transform is a popular method for skew detection. It is capable of locating fragmented lines in a binary image. Therefore given a group of black pixels, one can find the imaginary line or lines that go through the maximum number of these pixels. Given a binary image with a dominant text area, the detected lines will most probably go along the whole middle zone of the textual lines. Hence these lines have approximately the same skew as the reference lines of the text which the skew of the whole page. Whenever the Hough transform is used, there is always a tradeoff between accuracy and speed. The more accurate the angles of the detected lines are, the more computation is required. In addition the computation time depends on the number of pixels in the image. The basic approach to implement of Hough Transform is as follow:

3.1 PSEUDO CODE: Algorithm 0 (Basic Approach)

1. Input :Image of **D1×D2** pixels each either **1** and **0**
2. Here **?á**can take any value between **0=â?áásqrt(D1^2 + D2^2).**
3. Quantize Parameter Space with steps **d?** and **d?.**
4. Construct hough matrix **H[?,?]**with all elements initialized **0.**
5. For each pixel, calculate **?** for differnt values of angle **?** and increment corresponding Hough matrix accordingly

```
for( i=1;i<=D1;i=i+1)
    {       for(j=1;j<= D2;j=j+1)
            {       if(IMAGE[i,j]= =0)   /* Pixel is l 0 for black, 1 for black*/
                    {       for(k=1;j<=45;k=k+ d?)
                            {       p=i*cos(k)+j*sin(k);
                                    H[p,k]=H[p,k]+1;
                            }
    }       }       }
```

6. Find the element of Hough matrix **H,** which has the maximum value. Say it is **H[m?¡ m?i].** This **m?¡ m?i**represents the most dominant line in Image. From **m?¡ m?È**we can find the orientation of image.

3.1.1 MATLAB Implementation of Algorithm 0

```
image = imread('imagefile.tif');

size1=size(image);
d1=size1(1);
d2=size1(2);
PMAX=sqrt(d1*d1+d2*d2);
PMAX=PMAX*5+1;
HTABLE1= zeros (PMAX, 450+1);

 for i=1:d1
 for j=1:d2
          if(image(i,j)==0)
               for a=0:.0.10:45
                        p=i*sin((a*pi)/180.0)+j*cos((a*pi)/180.0);
                         p1=round(p*5)+1;
                          HTABLE1(p1,10*a+1)=HTABLE1(p1,10*a+1)+1;
                end
          end
   end
   end

maxm=max(HTABLE1);
maxv=max(max(HTABLE1));
```

```
angle=find(maxm==maxv);
angle=(angle-1)*.10;
fprintf('\nangle is  %6.2f',angle);
```

3.1.2 Assumptions In Above Code

1. Document image is bitmap i.e. one pixel need one bit for storage and two colors are possible, white for background (value=1) and black for foreground(value=0)
2. Document image Contains only Text
3. Format of Document Image is **.tif**
4. Document Image is in Uncompressed Format
5. Angle (**?**) can range from 0 TO 45 degrees

3.1.3 Important Points about above Implementation

HTABLE1 is matrix which is used to store the votes for the corresponding (**?,?**) pair. Here size of the HTABLE1 solely depends upon the accuracy we need. So level of accuracy we need directly depends upon the size of this Hough Table. In the above code, we are having step size of 0.10, so we are having accuracy level up to one decimal place.

3.1.4 Working of the above Code

For each foreground black pixel(image(i,j)=0), following loop will run

```
for a=0:.0.10:45
        p=j*sin((a*pi)/180.0)+i*cos((a*pi)/180.0);
         p1=round(p*5)+1;
          HTABLE1(p1,10*a+1)=HTABLE1(p1,10*a+1)+1;
end
```

Initially all entries in the htable1 will be zeros. First time this loop will run, the htable1 may have data like below

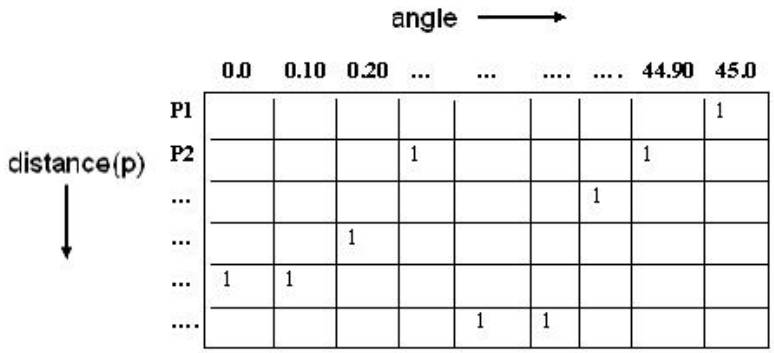


Figure 3.1: Hough Table (htable1) after voting of first foreground pixel

Second time this loop run for second black pixel, It would have data like below

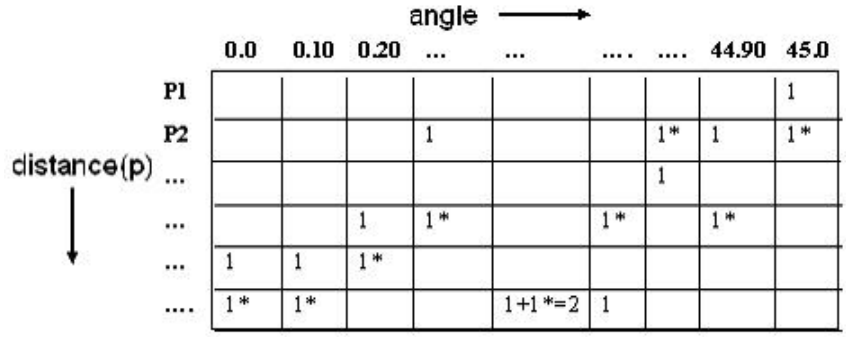


Figure 3.2: Hough Table (htable1) after voting of second foreground pixel

Similarly this loop will run for third black pixel and so on. Suppose at the end of last black pixel, the htable matrix looks like that

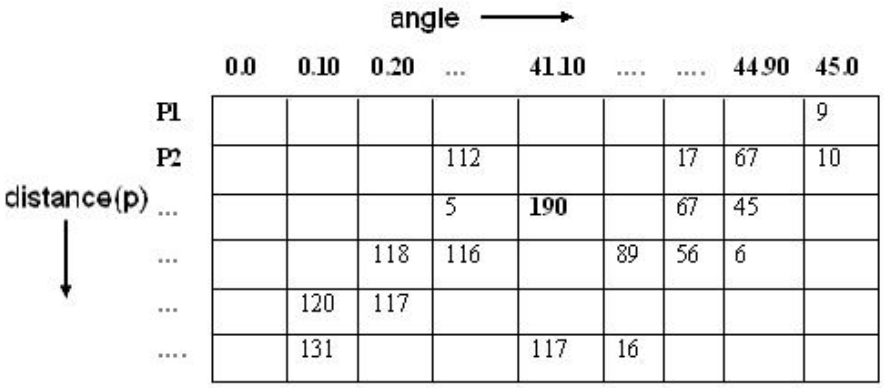


Figure 3.3: Hough Table (htable1) after voting of last foreground pixel

Now maximum value in the htable matrix is **190,** which is corresponding to angle 41.10 degrees. So value of **?** give by Hough Transform is **41.1^0.**

3.2 Problem Definition

Space Problem Suppose we need to calculate **?** accurate up to one decimal place and angle can vary from 0 to 50 degrees. Further suppose we have descretize the **?jin PMAX** Then size of htable will be **PMAX×450 approx**.

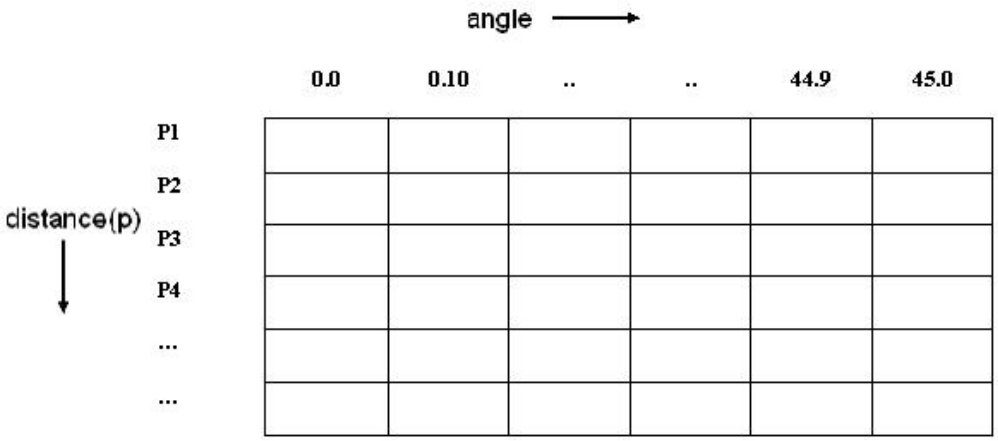


Figure 8.4 Hough Table (Htable1) For Accuracy Of One Decimal Place

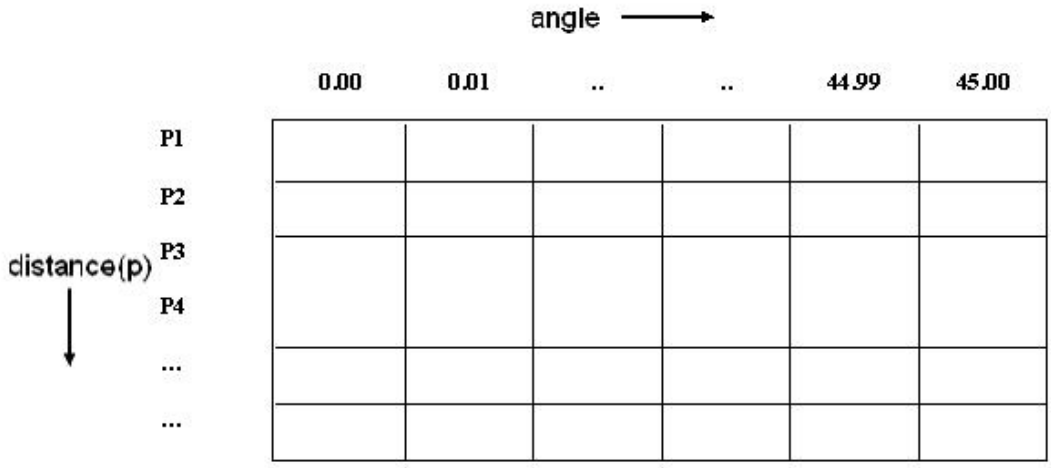


Figure 3.5 Hough Table (Htable1) For Accuracy Of Two Decimal Place

If we need accuracy level of angle up to two decimal places then size of matrix **Hough Table** will be **PMAX×4500 approx** as shown below in diagram**.** So it requires lot of memory.

Similarly if we need accurate angle up to level of three decimal places then above implementation will need htable of size **PMAX×45000.**

Time Problem: Above Algorithm consists of following segment

```
for a=0:.0.10:45
        p=j*sin((a*pi)/180.0)+i*cos((a*pi)/180.0);
        p1=round(p*5)+1;
        htable1(p1,10*a+1)=htable1(p1,10*a+1)+1;
end
```

In the first statement of above code segment (**for a=0:.0.10:45**), 0.10 is step size. So above loop will run from a=0 to a=45 and incrementing a by 0.10 in each step. So it will run for **451 times** for single pixel whose value is 0 (black pixel).

If we want accuracy level for angle up to two decimal places then this loop will run for **4501 times** for single foreground pixel.

In the next Chapter, solution of this problem has been provided in the Algorithm1 and Algorithm2.

CHAPTER 4
PROPOSED SOLUTION

4.1 ALGORITHM 1

Major problem with algorithm 0 is that its accuracy level depends upon how much fine the step size for the angle we uses. In algorithm 0, we were using data structure HTABLE1. Suppose we are having angle range from 0 to 44.9 degrees and we want results accurate up to one decimal place. Then we require HTABLE 1of size PMAX×450 approx. Now we can modify algorithm 0 to use space for Hough table as PMAX×90 (say it HTABLE2), which is as 5^{th} part of original HTABLE1 in size. Algorithm1 is designed to use to reduce the space requirement of algorithm0. However it takes more time to run. But using this approach we can run it on system having space constraint or in small memory models.

14.1.1 PSEUDO CODE: Algorithm1

Assumptions

Following assumptions have been used in pseudo code.

1. Angle can range from 0 to 44.9 degrees:
2. Accuracy up to one decimal place is required.
3. Hough table (HTABLE2) has size PMAX×90
4. ARRAYMAX is1D array of size 450.
5. variable INDEXA is used for indexing ARRAYMAX

Pseudo Code

7. Input :Image of **D1×D2** pixels each either **1** and **0**
8. INDEXA=0, HTABLE2=zeros(PMAX,90)
3. for(ang=0;ang<= 44.9;ang=ang+0.10)

```
2.1      for( i=1 to D1) {
         for(j=1 to D2) {
                  if(IMAGE[i,j]= =0)   {
                     p=i*cos(ang)+j*sin(ang);
                    HTABLE2[p,ang*10] =HTABLE2[p,ang*10]+1;
                                        }

2.2      if(  (ang*10+1)%90=0)
         {
            2.2.1   for (k=1;k<=90;k++)
                    {
                ARRAYMAX(INDEXA) =max_of_col(HTABLE2, k);
                INDEXA=INDEXA+1;
                    }

            2.2.3          HTABLE2=zeros(PMAX,90);
```

9. Find the index of element of arraymax which has highest value.
   Maximumval=index_of_max(ARRAYMAX);.
   Angle=maximumval/10;
10. END

14.1.2 Description of Functions Used In Above Pseudo Code

Zeros( ) This function is used to create matrix containing zeros. Its syntax is

Matrix=Zeros (dimension1, dimension2)

Max_of_col( ) This function takes two arguments. First argument is matrix and second is scalar number which represents column number of that matrix and returns maximum value in matrix in that column

```
Integer max_of_col( integer matrix, integer column)
{
max=matrix[1,column];
for(i=1;i<=matrix.dimension1;i++)
{
if(max<matrix[I,column]);
max=matrix[I,column];
}
return max;
}
```

index_of_max( ) This function takes one dimensional array as argument and returns index of largest value.

```
Integer index_of_max(integer A[])
        {
        k=1;
        max=A[1];
        for(i=1;i<=A.length;i++)
                {
                if(max<A[i])
                        {
                        max=A[i];
                        k=i
                        }

                }
return k;
        }
```

14.1.3 Working Example of Algorithm 1

1. Take one 1D array say ARRAYMAX[450]

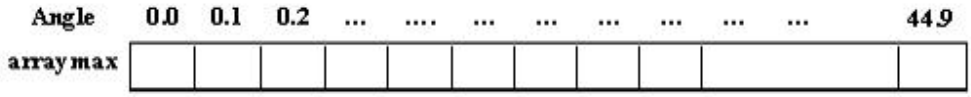


Figure 4.1 One Dimensional array arraymax

This array will be used to store the maximum vote for particular angle.

2. Take Hough table matrix say HTABLE2 whose size is PMAX×90

3. We will run basic Hough transform algorithm to fill the data for HTABLE2 [PMAX×90] for angle 0 to 8.9.

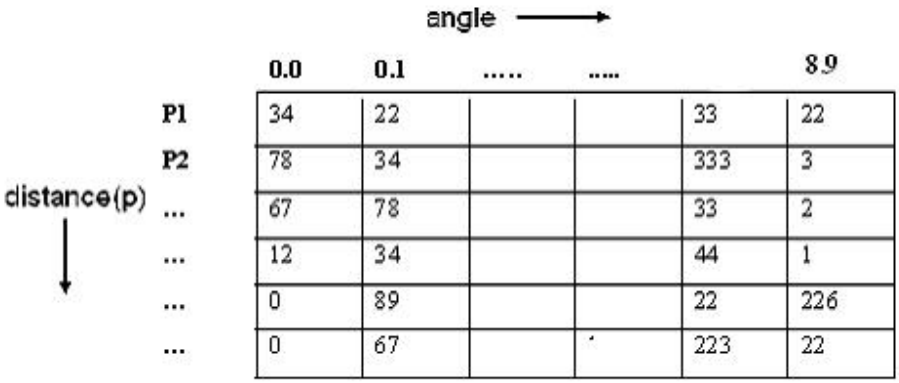


Figure 4.2 Hough Table after step 3

4. For each column of HTABLE2, find max value and copy it to corresponding ARRAYMAX array.

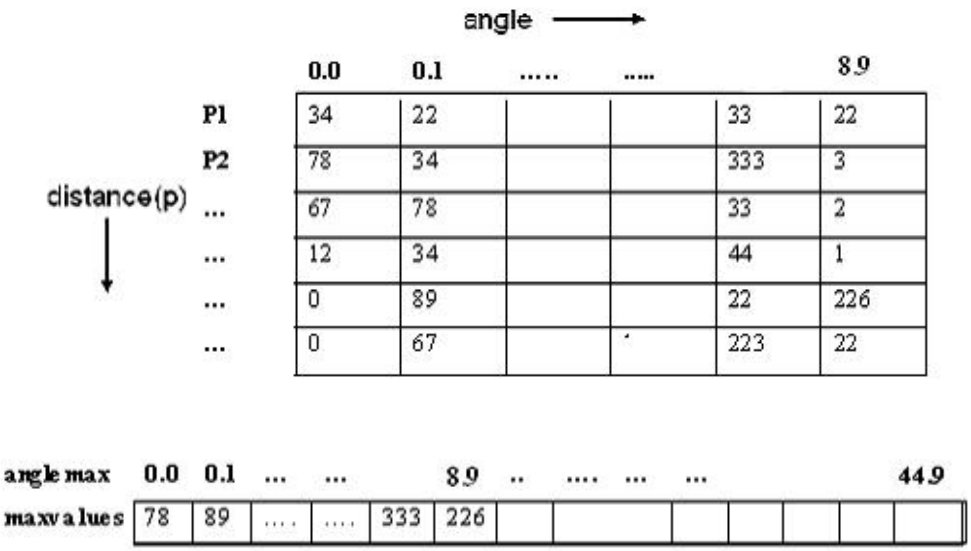


Figure 4.3 values **of** arraymax corresponding to Hough Table

5. Make the Hough table HTABLE2 empty by filling it with zeros.
6. Run the basic Hough algorithm again for angle 9.0 to 17.9. And use the same HTABLE2 to fill the voting data.

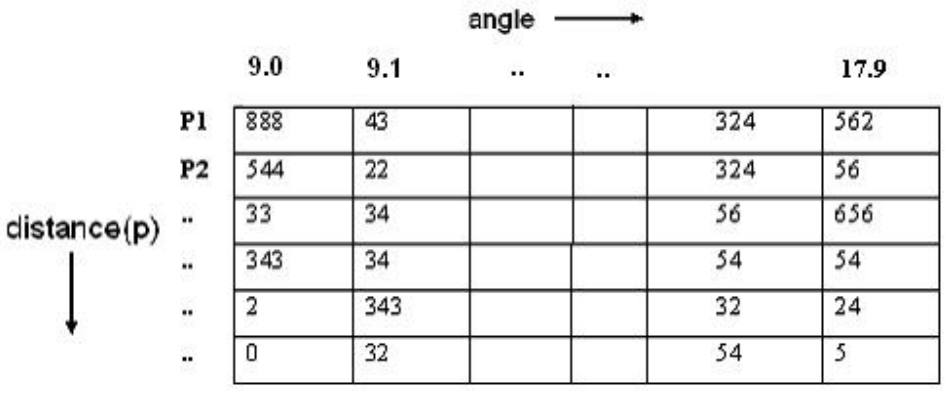


Figure 4.4 Hough Table after step 6

7. For each column of HTABLE2, find max value and copy it to corresponding ARRAYMAX array. ARRAYMAX will have following data after this step

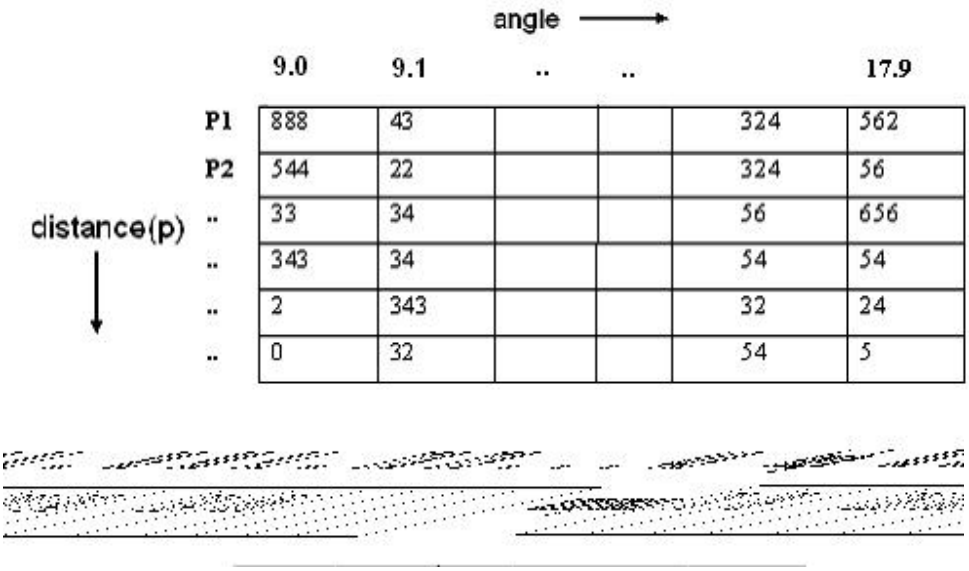


Figure 4.5 values o**f** arraymax corresponding to Hough Table after step 7

8. Repeat these steps (7) and(8) 3 times more and we will get ARRAYMAX as follow:

Figure 4.6 values o**f** arraymax corresponding to Hough Table

9. Find the angle which got the highest vote. In above case highest votes are 901 and corresponding angle is 22.2 degrees

`

14.1.4 MATLAB IMPLEMENTATION OF ALGORITHM 1

```
image=imread('imagefile.tif');
size1=size(image);
```

```
d1=size1(1);
d2=size1(2);
PMAX=sqrt(d1*d1+d2*d2)*5+1;
HTABLE2= zeros (PMAX, 90);
anglemax=zeros(1,450);
indexa=1;
for a=0:.10:45
                    for i=1:d1
                        for j=1:d2
                            if(image(i,j)==0)
                                p=i*sin((a*pi)/180.0)+j*cos((a*pi)/180.0);
                                p1=round(p*5)+1;
                                a1=rem(a*10,90)+1;
                                HTABLE2(p1,a1)=HTABLE2(p1,a1)+1;
                            end
                        end
                    end

                if(rem(a*10+1,90)==0)
                                ht=max(HTABLE2);
                                for k=1:100
                                    val=ht(k);
                                    anglemax(indexa)=val;
                                    indexa=indexa+1;
                                end
                                HTABLE2=zeros(PMAX,90);
                        end
 end
max_val=max(arraymax);
angle=find(arraymax = = max_val)/10;
fprintf('\nangle is  %f',angle);
```

Suppose we have to determine angle(**?**) accurate up to 1 decimal places and and angle **(?)** can range from 0 to 49.9 degrees and distance (**?**)È can take PMAX number of values. Then algorithm 0 will take PMAX×450 space for Hough Table, but algorithm 1 will take PMAX×90 for hough table. Also it uses one more array ARRAYMAX of size 500. So space requirement is reduced approximately by factor of 5. The amount of space we can save depends upon programmer choice. Size of Hough Table can be theoretically reduced to PMAX×1, but it will take lot of time to execute. So algorithm 0 and algorithm 1 are perfect example of time and space trade off. Reason of increase of computational time is that we have to clear the Hough Table after one pass to reuse it as shown in example above. Also we require additional task of storing the maximum value of each column in ARRAYMAX.

14.2 ALGORITHM 2

Algorithm 0 suffers from space and time problem. Algorithm solves first problem. But time problem becomes even vigorous in algorithm 1. So we need more efficient solution that optimizes space as well as time. In this section we have provided algorithm that attempts to achieve both goal that is reducing space and time complexities. Algorithm 0 takes Hough table of size PMAX×450 approx and Algorithm1 takes Hough Table of size PMAX×90. Algorithm 2 will take even smaller Hough Table. Its size is PMAX×45.

14.2.1 BASIC IDEA

Basic Idea of Algorithm is very simple and straight forward. It starts with calculation of angle accurate up to zero decimal place using basic algorithm. Suppose calculated angle is 23 degree. Then run again basic algorithm from 22 degrees to 24 degrees with accuracy of one decimal place. Suppose angle determined is 22.3 degrees. Suppose we want more accuracy, then we will run basic algorithm again from 22.2 to 22.4 with accuracy of two decimal places. Suppose resultant angle comes out to be 22.28 degrees. Same process can be iterated till level of desired accuracy is achieved.

14.2.2 PSEUDO CODE: Algorithm2

Assumptions

1. Angle can range from 0 to 45 degrees.
2. Accuracy up to one decimal place is required.
3. Hough table (HTABLE3) has size PMAX×45
4. INDEX_ANG is used for indexing the angle dimension in HTABLE3.

Pseudo code

1. Input :Image of **D1×D2** pixels each either **1** and **0**
2. precision=1, start_angle=0, end_angle=45, step=1
3. while(percision=0.1)

```
3.1 HTABLE3= zeros(PMAX,50);
3.2 for(i=1:i=d1;i++)
        for(j=1:j=d2;j++)
            if(IMAGE[i,j]=0)
            {
            INDEX_ANG=1;
             for(ang=start_angle; ang =<end_angle; ang=ang+step)
                    {
                       p=j*sin((ang*pi)/180.0)+i*cos((ang*pi)/180.0);
                       p1=round(p*5)+1;
                       HTABLE3(p1, INDEX_ANG)=HTABLE3(p1,
                    INDEX_ANG)+1;
                       INDEX_ANG = INDEX_ANG +1;
                    }
              }
```

3.3 Find the index of element of HTABLE3 which has highest value.

maxind=index_of_max(HTABLE3);.

3.4 angle= step*(maxind-1)+start

3.5 start_angle=angle-step

end_angle=angle+step

step=step/10.0

percision=percision/10.0

4. Print: angle
5. End

In above algorithm, variable **precision** is used for accuracy level. If we need accuracy level of one decimal place, then it is checked internally for greater than and equal to 0.1 or until it becomes 0.1. Initially value for it is 1. Variable **step** is used as incrementing step in angle. In first iteration, step size is 1. it means we will start from start angle (0) , and incrementing it by 1 (step size) till we reach end angle. In second iteration, step will be divided by 10. its value will be 0.10. and variable **ang** will be incremented by 0.10 starting from start angle to end angle

Number of Iterations depends upon the accuracy level. If we need angle accurate up to one decimal place, then number of iterations will be two as in above case. So number of iterations will be equals to number of decimal places plus one.

This algorithm is iterative and more efficient in terms of space and time.

14.2.3 Working Example of Algorithm 2

Suppose we have to determine angle (**?**) accurate up to one decimal places and angle **(?)** can range from 0 to 44.9 degrees and distance (**?**) can take PMAX number of values. This algorithm will use Hough Table HTABLE3 of size PMAX×45

1. Take one Hough Table HTABLE23 of size PMAX×45

2. Start with Step size of one i.e. STEP=1.
3. We will run basic Hough transform algorithm to fill the data for HTABLE3 [PMAX×45] for angle 0 to 45 using step size for angle equals to one. Here start angle =0 and end angle =45.

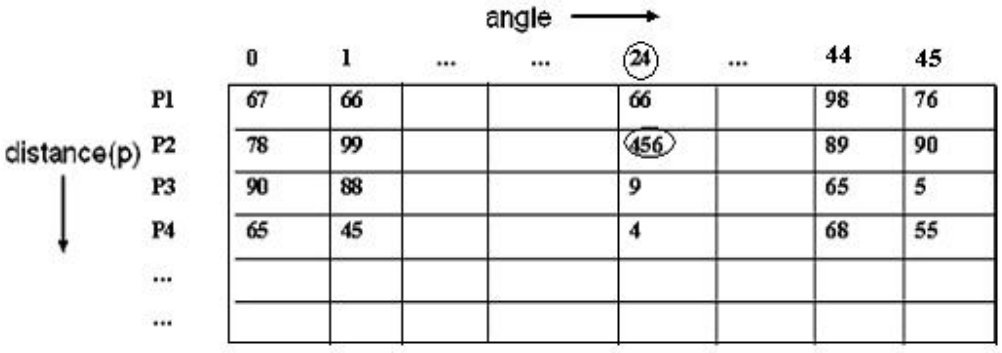


Figure 4.7 Hough Table after step 3

Angle corresponding to maximum value (456) in HTABLE is 24 degree.
Angle=24 degree

4. As step size was 1. and determined angle is 24. so decrement and increment the determined angle by step to find new interval for angle calculation.

Start angle =angle-step=24-1 =23
end angle = angle+step=24+1=25
now search interval for angle will be from 23 to 25.
Step=step/10=0.10

5. Clear the Hough Table to zeros.

Run basic Hough transform algorithm to fill the data for HTABLE3 for angle 0 to 50 using step size for angle equals to 0.10. Here start angle =23 and end angle

=25.

angle →

distance(p) ↓	23	23.1	23.2	……	23.9	…	24.9	25.0
P1	67	88	23		53		34	56
P2	45	76	43		34		56	56
P3	55	78	67		**452**		67	45
…								
…								
…								

Figure 4.8 Hough Table after step 5

Angle corresponding to maximum value (452) in HTABLE is 23.9 degree.

6. Repeat the same procedure to find the angle accurate up to more decimal places

14.2.4 MATLAB IMPLEMENTATION OF ALGORITHM 2

```
image=imread('imagefile.tif');
size1=size(image);
d1=size1(1);
d2=size1(2);
PMAX=sqrt(d1*d1+d2*d2)*5+1;

percision=1;                /*for precision*/
start_angle=0;
end_angle =45;
step=1;
while ge(percision,0.1)
HTABLE3= zeros (PMAX, 45);
        size1=size(image);
        d1=size1(1);
```

```
        d2=size1(2);
        ind=1;

        for i=1:d1
                for j=1:d2
                    if(image(i,j)==0)
                            ind_ang=1;
                            for a=start_angle:step:end_angle

                                    p=j*sin((a*pi)/180.0)+i*cos((a*pi)/180.0);
                                    p1=round(p*5)+1;
                                    HTABLE3(p1,ind_ang)=HTABLE3(p1,ind_ang)+1;
                                     ind_ang=ind_ang+1;



                                    end
                            end
                 end
        end

        maxm=max(HTABLE3);
        maxv=max(max(HTABLE3));
        maxind=find(maxm==maxv);
        angle=step*(maxind-1)+start;
        start_angle=angle-step;
        end_angle=angle+step;
        step=step/10.0;
        percision=percision/10.0;

end
angle=mean(angle);
```

fprintf('\nangle is %f ',angle);

14.2.5 SPACE AND TIME REQUIREMENT FOR ALGORITHM2

Algorithm1 was better than algorithm0 in terms of space but at the expense of time. Algoritm2 optimizes the space and time. It is iterative in nature. In each iteration its accuracy level is increased by one decimal place. Suppose

a) Accuracy up to 2 decimal places is needed for angle.
b) Angle can range from 0 to 45 degrees
c) Distance (**?**) can have PMAX different values

Space:

(a) Algorithm2 will use Hough Table HTABLE3 of size PMAX ×45, whereas algorithm0 was using HTABLE1 of size PMAX ×450.
(b) Space Requirement for Algorithm0 depends upon the accuracy level. Accuracy up to more decimal places we need, and then bigger space we need for storing Hough Table. Algorithm 2 is **INDEPENDENT** of accuracy level. More the accuracy we need, more the iterations will be carried out without need of more memory space.

Time:

Algorithm 0 consists of following code segment

```
for a=0:.0.10:45
        p=j*sin((a*pi)/180.0)+i*cos((a*pi)/180.0);
         p1=round(p*5)+1;
          HTABLE1(p1,10*a+1)=HTABLE1(p1,10*a+1)+1;
end
```

In the first statement of above code segment (**for a=0:.0.10:45**), 0.10 is step size. So above loop will run from a=0 to a=45 and incrementing a by 0.10 in each step. So it will run for 4**50 times approx** for single pixel whose value is 0 (black pixel). But if we run

Algorithm2, then it will consists of two iterations as number of iterations is one more than the number of decimal places up to which accuracy in angle is needed.

1. Loop will run for 45 times in iteration 1
2. Loop will run for 21 times in iteration 1

So totally 45+21=67 times it will run. Comparing it with algorithm 0, which was being run this block of code for 450 times?

If we want accuracy level for angle up to two decimal places then Algorithm 0 loop will run for **4500 times approx** for single foreground pixel. But algorithm 2 will run for 45+21+21 =87 times.

CHAPTER 5
CONCLUSION

In Chapter 3, we discussed the basic algorithm for implementing Hough Transform "**Algorithm 0**". It was having space complexity

S = O (PMAX ×ANGLE ×(D×10)) **…….1**

Where PMAX is different values for distance **(?)** parameter.
ANGLE is maximum angle for **(?)**
D (=1) is number of decimal places up to which accurate angle is needed.

Suppose we have to determine angle(**?**) accurate up to 1 decimal places and and angle **(?)** can range from 0 to 45 degrees and distance (**?**) can take PMAX number of values. Then algorithm 0 will take PMAX×450 approx space for Hough Table. Similarly if we need angle(**?**) accurate up to 2 decimal places, then algorithm 0 will take PMAX×4500 space for Hough Table. So size of Hough Table is directly dependent of accuracy level **(D).**

Time complexity of algorithm0 is given by

T = O (n×ANGLE × (D×10)) **……2**

Where **n** is total no of foreground pixels
ANGLE is maximum angle for **(?)**
D is number of decimal places up to which accurate angle is needed.

Putting **ANGLE × (D×10) = A** in (2) we get

T = O (n×A) **……3**

Here **A** is number of different values for angle.

If we have to determine angle(**?¹**) accurate up to 1 decimal places and and angle **(?¹)** can range from 0 to 45 degrees. Numbers of foreground pixels are **n,** then time taken by algorithm 0 will be **(n×450).** And if we need accuracy level of 2 decimal places, then algorithm will have time complexity of **n×4500.** By this figure we mean that following code within bounded rectangle will run for **n×4500 times.**

```
for a=0:.0.01:45
        p=j*sin((a*pi)/180.0)+i*cos((a*pi)/180.0);
        p1=round(p*5)+1;
        HTABLE1(p1,100*a+1)=HTABLE1(p1,100*a+1)+1;
end
```

In chapter 4, **Algorithm 1** is proposed, which attempts to reduce the memory requirement for algorithm 0. It will take PMAX×90 for Hough table for accuracy of one decimal place. Also it uses one more array ARRAYMAX of size 450. So space requirement is reduced approximately by factor of 5. The amount of space we can save depends upon programmer choice. Size of Hough Table can be theoretically reduced to PMAX×1, but it will take lot of time to execute. So algorithm 0 and algorithm 1 are perfect example of time and space trade off. Reason of increase of computational time is that we have to copy clear the Hough Table after one pass to reuse it as shown in example above. Also we require additional task of storing the maximum value of each column in ARRAYMAX.

Algorithm 1 solves the problem of space, but at the cost of extra time and computations.

In chapter 4, "**Algorithm 2"**is proposed for solving this problem. Algorithm2 is better than both algorithm0 and algorith1 in terms of space and memory. Algorithm1 was better than algorithm0 in terms of space but at the expense of time. Algoritm2 optimizes the space and time. It is iterative in nature. In each iteration, its accuracy level is

increased by one decimal place. To compare this algorithm0, algorithm1 and algorithm2, we are having following assumptions.

Assumptions

d) Accuracy up to 1 decimal places is needed for angle.
e) Angle can range from 0 to 45 degrees
f) Distance (**?á**) can have PMAX different values
g) No of foreground pixels are **n.**

1. Algorithm 0 will use Hough table HTABLE1 of size PMAX×450 approx, where as Algorithm1 will use Hough Table HTABLE2 of size PMAX ×90 and algorithm1 will use Hough Table HTABLE3 of size PMAX ×45.

SR. NO.	PARAMETER	ALGORITM 0	ALGORITM 1	ALGORITM 2
1	**Space**	Highest (PMAX×450)	Lesser than Algorithm0 (PMAX×90 and can be reduced to PMAX×1)	Far lesser than Algorithm 0 (PMAX×45)
2	**Time**	High (**n** ×4500)	More than Algorithm 1	Least (**n**×66)
3	**Iterative Nature**	Non-Iterative	Non-Iterative	Iterative
4	**Dependence of space requirement on accuracy lvl.**	Yes	…..	No

Table 5.9: Comparison of Algorithm 0, 1, 2

2. Space Requirement for Algorithm0 depends upon the accuracy level. Accuracy up to more decimal places we need, and then bigger space we need for storing Hough Table. Space requirement for Algorithm can be reduced to PMAX ×1. It depends upon programmer how big Hough Table he uses. In our example in chapter 4, we used the Hough Table of size PMAX ×90. Algorithm 2 is **INDEPENDENT** of accuracy level. More the accuracy we need, more the iterations will be carried out without need of more memory space.
3. Algorithm 1 takes more time than algorithm 0. but algorithm 2 most efficient in terms of time as well as space. For algorithm 2 basic code will run as follow
    a. Loop will run for 45 times in iteration 1
    b. Loop will run for 21 times in iteration 2

Loop will run for 45+21=66 times for algorithm2 as comapared to 4500 times for algorithm 0.

4. Algorithm 0 and Algorithm 1 are non-iterative whereas Algoritm 2 is Iterative in nature.

Hence Algorithm 2 is better than algorithm0 and algorithm1. Algorithm 2 has scope of further improvements. Algorithm2 can be combined to algorithm1 to reduce the more space.

REFERENCES

.

[1] Duda, R. O. and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," Comm. ACM, Vol. 15, pp. 11–15 (January, 1972),

[2] M. Sarfraz, S.A. Mahmoud and Z. Rasheed,On Skew Estimation and Correction of Text, Computer Graphics, Imaging and Visualisation(CGIV 2007) 0-7695-2928-3/07

[3] Akiyama, T. and Hagita, N. (1990), Automatic entry system for printed documents, Pattern Recognition, Vol. 23(11), 1141 – 1154, Elsevier Science.

[4] Postl, W., (1986). Detection of linear oblique structures and skew scan in digitized documents. In: Proc. 8th Internat. Conf. on Pattern Recognition, Paris, France, 687–689.

[5] Bloomberg, D.S., Kopec, G.E., (1993). Method and apparatus for identification and correction of document skew. Xerox Corporation, US Patent 5,187,753.

[6] Bloomberg, D.S., Kopec, G.E., Dasari, L., (1995). Measuring document image skew and orientation. Proc. SPIE 2422, 302–316.

[7] Ishitani, Y,(1993), Document skew detection based on local region complexity. In: Proc. 2nd Internat. Conf. on Document Analysis and Recognition, Tsukuba Science City, Japan, 49–52.

[8] Liolios, N., Fakotakis, N., Kokkinakis, G., 2002. On the generalization of the form identification and skew detection problem. Pattern Recognition 35, 253–264.

[9] Hashizume A, Yeh P-S, Rosenfeld A. A method of detecting the orientation of aligned components. Pattern Recognition Letters 1986; 4:125–132

10. Liu J, Lee C-M, Shu R-B. An efficient method for the skew normalization of a document image. 11th International Conference on Pattern Recognition ICPR 1992: 152–155

11. www.cis.hut.fi/~jhollmen/dippa/node30.html accessed on 24/5/2008

12. Smith, L.I, A tutorial on Principal Components Analysis; February 26, 2002.

13 Steinherz, T. Intrator N. and Rivlin, E. (1999), Skew detection via principal components analysis. In Proceedings of the 5th International Conference on Document Analysis and Recognition, pages 153–156. IEEE Computer Society.

[14] S. C. Hinds, J. L. Fisher, and D. P. D'Amato. A document skew detection method using runlength encoding and the Hough transform. In Proceedings Int. Conf. on Pattern Recognition, pages 464{468, June 1990.

[15] D. S. Le, G. R. Thoma, and H. Wechsler. Automated page orientation and skew angle detection for binary document images. Pattern Recognition, 27(10):1325{1344, 1994.

[16] B. Yu and A. K. Jain. A robust and fast skew detection algorithm for generic documents. Pattern Recognition, 29(10):1599{1630, 1996.

[17] Yu, B., Jain, A.K., 1996. A robust and fast skew detection algorithm for generic documents. Pattern Recognition 29 (10), 1599–1629.

[18] Yan, H., 1993. Skew correction of document images using inerline cross-correlation. Comput. Vision Graphics Image Process. 55 (6), 538–543.

[19] A Amin and S. Fischer, A Document Skew Detection Method Using the Hough Transform, Pattern Analysis & Applications (2000)3:243-253

[20] Amin A, Fischer S. Fast algorithm for skew detection. IS&T/SPIE Conference on Real-Time Imaging, USA, 1996; 65–76

[21] Manjunath Aradhya V N, Hemantha Kumar G, and Shivakumara P, Skew Detection Technique for Binary Document Images based on Hough Transform, International Journal of Information Technology Volume 3 Number 3

[22] Walpole RE, Myres RH. Probability and Statistics for Engineers and Scientists, 4th ed. Macmillan, 1990: 362

[23] Paeth A. A fast algorithm for general raster rotation. Proceedings Graphics Interface Vision Interface 1986: 77–81

[24] Ahmed Maher and Ward Rabab, A Rotation Invariant rule-Based Thinning Algorithm for Character Recognition, IEEE Transaction on pattern analysis and machine Intelligence, Vol 24, No 12, December 2002

[25] Steinherz T, Intrator N and Rivlin E , Skew Detection via Principal Components Analysis, Fifth International Conference on Document Analysis and Recognition, 1999. ICDAR '99.

[26] Lehal G S and Dhir R, A Range Free Skew Detection Technique for Digitized Gurmukhi Script Documents, Fifth International Conference on Document Analysis and Recognition, 1999. ICDAR '99.

[27] Baird HS., Anatomy of a versatile page reader. Proceedings of the IEEE, 1992; 80(7):1059–1065

[28] Hinds SC, Fisher JL, D'Amato DP. A Document Skew Detection Method Using Run-Length Encoding And The Hough Transform. Proceedings 10th International Conference On Pattern Recognition 1990; 464–468

[29] O’Gorman L. The Document Spectrum For Page Layout Analysis. IEEE Transactions On Pattern Analysis And Machine Intelligence 1993; 15(11):1162–1173

[30] Postl W. Detection Of Linear Oblique Structures And Skew Scan In Digitized Documents. Proceedings 8th International Conference On Pattern Recognition 1986: 687–689

LIST OF PAPERS PUBLISHED/ACCEPTED

[1] Arun Arora, Amrit Kaur **"**Review of Skew Detection Algorithms using Hough Transform for Document Image" in "Apeejay Journal of Computer Science and Applications", 2008 **(Status: Accepted)**

[2] Arun Arora, Sandeep Sharma "Skew Detection Using Hough Transformation" in "4th International Conference on Challenges & Developments in IT (ICCDIT-2008)", 2008 **(Status: Accepted)**