A Fast and Memory Efficient Algorithm for Skew Detection in Document Images

Nitin Bhatia
(Department of Computer Science, DAV College, Jalandhar 144008, INDIA
Email: n_bhatia78@yahoo.com)

ABSTRACT
A fast, memory efficient and fully automated technique called hierarchical Hough transform (HHT), for skew detection in document images is developed. The technique is based on simple Hough transform (SHT) which adopts a new approach for finding maximum votes in Hough parameter space by recursively dividing the parameter space into quadrants and reducing the search by removing those quadrants which receive votes less than a threshold value. A combination of quad tree and binary tree data structures is used to implement the concept and a depth-first search technique is used to reduce the memory requirement. An automated thresholding mechanism is designed making the algorithm independent. Since the parameter space is rectangular and the leaf node is a single pixel, the accuracy of the HHT algorithm is the similar to SHT. The global optimum is reached very fast as only a few nodes, of the quad tree or binary tree, are searched.

Keywords *-Hough transform, line detection, optical character recognition, skew detection.*

I. INTRODUCTION

Skew detection is an important step in document image analysis. It is an integral part of optical character recognition (OCR) system which is an indispensable tool in office automation, automatic reading of postal addresses, forms, government records, credit card imprints, postal mail, etc. An OCR system converts an image form of text to editable form, thus facilitating the process of modifying the document through text editors. The editable form of the document requires much less space compared to the space required by the image form. Prior to the conversion process, a document is scanned and saved in a file. It is during the scanning process that the document is skewed inadvertently.

The method of skew detection falls broadly into three categories: Hough transform, projection profile and nearest neighbor. O' Gorman [1] has discussed a detailed account of these categories. These methods and a few other hybrid methods have been described in [2-6]. Techniques based on Hough transform [12, 13] are very popular because of the high accuracy, simplicity and ability of the transform to provide satisfactory results even under noisy conditions. However, the major drawbacks of the technique are its high memory requirement and low speed. The time complexity of Hough transform is $O(N_b L)$ where N_b is the number of image pixels and L is the resolution of the slope parameter which is normally varied from $-90°$ to $+90°$ with an interval of $1°$. Higher resolution is achieved by increasing the value of L. The number of image pixels, N_b, is normally very large, sometimes of the order of 10^6 or higher for an $A4$ size document scanned at 300 dpi or more. The space complexity is characterized by a 2-D array, called "accumulator" which needs $3N_b L$ number of elements under normal implementation of the technique. Several researchers have modified Hough transform to accelerate the speed and reduce memory requirements. The fast Hough transform (FHT) introduced by Li et al. [7] uses a quad tree decomposition of a square parameter space recursively. The method proceeds by subdividing quadrants which receive more than a specified number of votes. The processing is continued until either the number of votes is above the threshold or the size of the quadrant is more than the minimum size. The adaptive Hough transform of Illingworth and Kittler [8] uses initially a coarse grid of the complete parameter space and later on only those cells are processed which are the potential candidates for further processing after performing thresholding and connected component analysis. The hierarchical Hough transform (HHT) of Princen et al. [9] uses a pyramidal structure applying HT type algorithm at all levels.

The concept of FHT [7] was later modified by Guil et al. [10] after eliminating some of the drawbacks of FHT and making it more computation efficient. However, its full potential is not realized because of several drawbacks. First it uses a square parameter space which is not the case as the parameter space is a rectangular one: $[-1,1]\times[-W, W+H]$ in slope-intercept form for any image with size $H \times W$. Secondly, false lines are

likely to be introduced because of the decision strategy adopted by [7, 10]. Thirdly, it uses floating point arithmetic for performing calculations. Perhaps it is due to these reasons that the method is not popular despite its inherent strength in certain applications.

At a glance, FHT appears to be very promising in some applications where the maximum peak is required to be determined. One of such applications is the determination of skew in document images. In this paper, we propose a fast decision technique employing the basic concept of FHT [7] and eliminate its drawbacks. We further explore the efficacy of the method to find global maximum and apply it to detect skew in document images. It is shown that the proposed method is very fast and determines skew as accurately as SHT and requires very less space for its execution.

II. SIMPLE HOUGH TRANSFORM

The Hough transform maps a line in spatial domain to a point in Hough parameter space. The Hough parameter space is a 2-D space and is described in the form (a,b) in Cartesian coordinates system where a is the slope and b is the intercept of the line passing through the point (x_0, y_0). The transform is

$$b = -ax_0 + y_0 \qquad (1)$$

Since the dynamic range of both parameters, a and b, is $[-\infty, \infty]$, it is restricted to finite domain by splitting the range of a in two parts: $|a| \le 1$ and $|a| > 1$, as given by following equation:

$$b = \begin{cases} -ax_0 + y_0, & |a| \le 1 \\ -ay_0 + x_0, & otherwise \end{cases} \qquad (2)$$

The parameter space $b = f(a)$, where $f(a)$ is the right hand side of Eq. (2) for $-1 \le a \le 1$, for an image whose resolution is $H \times W$, where H is image height (along vertical direction) and W is the image width (along horizontal direction). It is clear from Fig. 2 that, at $a = 0$, $b_{\min} = 0$ and $b_{\max} = H$, at $a = 1$, $b_{\min} = -W$ and $b_{\max} = H$, and at $a = -1$, $b_{\min} = 0$, $b_{\max} = W + H$. Thus, in the region $-1 \le a \le 1$, b spans from $-W$ to $W + H$.

1.1 Complexity of Hough Transform: The SHT works as follows: Let (x_0, y_0) be a point in the image space. We map it to (a_i, b_i) in Hough space by taking $a = a_i$ in the interval $[-1,1]$ and computing for b_i by using Eq. (2). The point (x_0, y_0), is thus said to vote for the cell (a_i, b_i) and the vote count, maintained by a 2-D array $P[b,a]$, called accumulator, is incremented by one. Initially, $P[b,a] = 0, \forall a$ and b. The parameters a and b take on the values: $a = 0,1,2,...,L-1$ and $b = 0,1,2,...,2W + H - 1$, respectively. We translate b by W to avoid b from being negative. Normally, the value of L is taken 90 each for both the parts of the slope, thus the total discrete values of a is 180. This is done so as to maintain the consistency with $\rho - \theta$ form of the transform where θ ranges from $-90°$ to $+90°$ normally with the interval of $1°$. The voting process is repeated for all image points and the accumulator is updated. At the end, we look for the values of a and b which provide the maximum value of $P[b,a]$, called the maximum votes. The value of parameters corresponding to maximum votes, say $a = a_0$ and $b = b_0$, are used to define the line which accumulates the maximum number of image points and is the line

$$y = a_0 x + b_0 \qquad (3)$$

The angle, $\theta = \tan^{-1}(a_0)$, is the skew.

It is clear from the above discussion that the time complexity of SHT is $O(N_b L)$ where N_b is the number of image pixels (throughout the paper we consider the binary documents and black pixels are treated as image pixels). The space complexity is $L(2W + H)$ words. It is clear from Fig. 2 that b lies between 0 and $W + H$ for $-1 \le a < 0$ and between $-W$ and H for $0 \le a \le 1$ and in order to make it to lie in the interval $[0, W + H]$ for $0 \le a \le 1$, we add W in the right hand side of Eq. (2), reducing space complexity to $L(W + H)$ words.

We now analyze the parameters that affect the time complexity which is very crucial for the efficient implementation of Hough transform. As mentioned

above, the time complexity depends on N_b and L. In slope-intercept space $\Delta\theta$ is not constant as

$$\Delta\theta = \tan^{-1}\left(\frac{\ell+1}{L}\right) - \tan^{-1}\left(\frac{\ell}{L}\right), \ell = 0,1,2,...,L-1.$$

Higher is the value of L, more precise will be the detection of skew. For $L = 180$, the skew is expected to be detected with an accuracy of $\pm\Delta\theta$, where

$$\Delta\theta = \tan^{-1}\left(\frac{\ell+1}{L}\right) - \tan^{-1}\left(\frac{\ell}{L}\right), \ell = 0,1,2,...,L-1.$$

Thus when the detection accuracy is enhanced from $\pm 1°$ to $\pm 0.5°$, the time taken is increased by a factor of two.

The effect of the parameter N_b on time complexity is much more pronounced than the effect of L, since N_b increases quadratically with respect to the size of documents and scanning density (dpi). That is why many researchers have taken much smaller values of N_b by using the techniques of skeletonization of characters or processing only a part of document or implementing block adjacency graph (BAG) [2, 11].

III. THE PROPOSED ALGORTHIM: MODIFIED HOUGH TRANSFORM (MHHT)

Let the size of the image be $H \times W$. The range of the parameter a is 0 to L, where L is normally taken as 180 but it is not restricted to a particular value. However, the range of b is fixed which is $-W$ to $W+H$. We, therefore, deal with a rectangular Hough space with opposite vertices $(0,-W)$ and $(L,W+H)$. In order to simplify the discussion, we consider a square image, $N \times N$, and the Hough space becomes the rectangle regions $(0,-N)\times(L,2N)$. The discussion is further restricted to the slope, a, for which $|a| \le 1$. The algorithm for the region $|a| > 1$ is a straight forward extension.

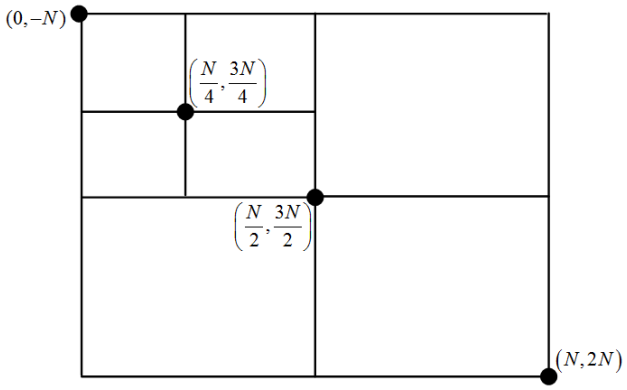

Fig. 1: The rectangular parameter space and its subdivisions.

The discrete form of Eq. (1) becomes,

$$b = \frac{-\ell}{N}x_0 + y_0, \ \ell = 0,1,2,...,N-1 \qquad (4)$$

The Hough space is divided recursively into rectangular quadrants as given in Fig.1, starting with the rectangle $(0,-N)\times(N,2N)$. All image pixels vote to this rectangle. We recursively divide it into four quadrants and decide about the votes for each of the four regions. The number of votes for a region is expected to be significantly less at each subdivision. This procedure can be implemented through a quad tree data structure in which the original rectangle represents the *'root'* node. A subdivision is called a *'child'* node. Since we consider rectangular Hough space of the size $N \times 3N$, the process of subdivision reaches a stage where the width of a rectangle is reduced to a single pixel but the length is more than one pixel. At this stage we use a binary tree by subdividing a node into two regions in the direction of the length and the process is repeated until the rectangle reduces to a unit square.

3.1 The voting Process: Let us assume that at a given stage of subdivision we decide about the votes for a node with apposite vertices (a_1,b_1) and (a_2,b_2). An image pixel (x_0,y_0) will vote for the node if it intersects the rectangle as shown in Fig. 2(b). When the node is below the line, Fig. 2(a), or above the line, Fig. 2(c), the pixel (x_0,y_0) does not vote for the node. The node will be below the line or above the line if the value of the expression, $f(a,b)$, given by

$$f(a,b) = b + ax_0 - y_0 \qquad (5)$$

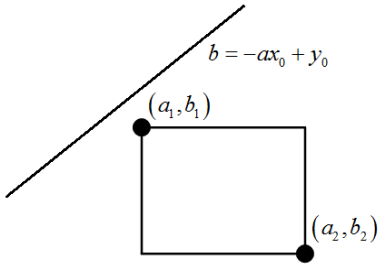

Fig. 2(a): Line is above the node.

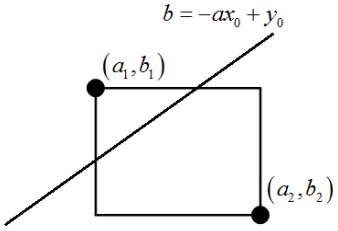

Fig. 2(a): Line intersects the node.

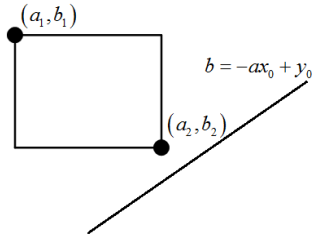

Fig. 2(a): Line is above the node.

has the same sign for all vertices of the node. In discrete integer form, $a = \frac{\ell}{N}$ and Eq. (5) assumes the form

$$f(\ell, b) = N(b - y_0) + \ell x_0 \qquad (6)$$

Thus, the testing criterion simply reduces to the evaluation of integer expression given in Eq. (6). This process is repeated for all image pixels of the parent node and a new voter list is created for the current node which is used for its *child* nodes. The process of subdivision and voting is carried on until the number of votes is above a threshold value or the size of a subdivision reduces to a single pixel.

3.2 Optimizing the Voting Process: The above decision making process is further optimized by reducing the number of calculations by resorting to one of the following two approaches.

3.3 First Approach: In the first approach the range of the slope parameter is divided into four parts: $-\infty < a < -1$, $-1 \le a < 0$, $0 \le a \le 1$ and $1 < a < \infty$. The expression, $f(\ell, b)$ given by Eq. (6) is evaluated for one or two vertices, instead of all four vertices.

The line represented by $b = -ax_0 + y_0$ passes from above the region if $f(\ell_1, b_1) = L(b_1 - y_0) + \ell_1 x_0 > 0$ and it passes from below the region if $f(\ell_2, b_2) = L(b_2 - y_0) + \ell_2 x_0 < 0$.

It is assumed that $\ell_2 \ge \ell_1$ and $b_2 \ge b_1$. This case is depicted in Fig. 3(a). It is clear that if $f(\ell, b)$ is positive for $(\ell, b) \equiv (\ell_1, b_1)$ then it will be positive for all other vertices and hence the line does not intersect the rectangle. Similarly, if $f(\ell, b)$ is negative for $(\ell, b) \equiv (\ell_2, b_2)$, then the function will be negative for all other vertices and again the line does not intersect the rectangle.

We need to evaluate function f only once (f_1) or twice $(f_1$ and $f_3)$. Assuming that the probability of f_1 or f_3 being positive or negative is $\frac{1}{2}$, we achieve a saving of 62.5% in the overall calculations as is apparent from the above discussion. Thus even if we include the overheads associated with decision statements, a minimum of 50% saving in execution time is likely to be achieved which makes the decision process faster by a factor of two.

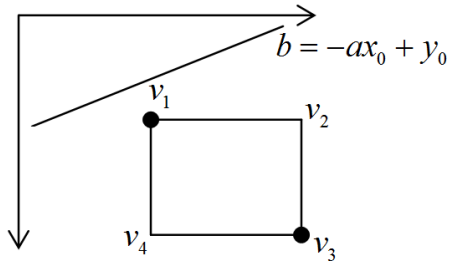

Fig. 3(a). Expression $b + ax_0 - y_0$ to beevaluated for v_1 and $v_3 (0 \le a \le 1)$

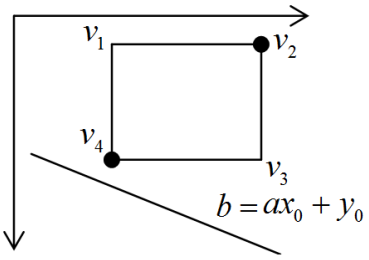

Fig. 3(b). Expression $b + ax_0 - y_0$ to be evaluated for v_2 and $v_4 (-1 \le a < 0)$

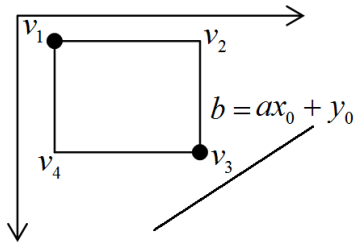

Fig. 3(c). Expression $b + ax_0 - y_0$ to be evaluated for v_1 and $v_3\,(1 < a < \infty)$

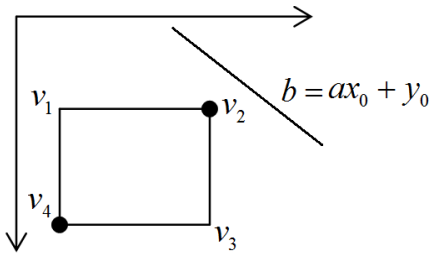

Fig. 3(d). Expression $b + ax_0 - y_0$ to be evaluated for v_2 and $v_4\,(-\infty < a < -1)$

Fig. 3(b), 3(c) and 3(d) depict the other three similar situations for the fast decision making process for the parts: $-1 \le a < 0$, $1 < a < \infty$, and $-\infty < a < -1$, respectively.

3.4 Second Approach: This approach requires the evaluation of two expressions on a scan line $y = y_0$ which provides two values of x, say x_1 and x_2. All image points (x_i, y_0), such that $x_1 \le x_i \le x_2$ vote for the node. The above approach can be made faster by adopting a special data structure for image pixels and by modifying the approach of decision making for the intersection of the line with a node.

We revert to the function, $f(\ell, b)$, given in first approach and conclude that an image point (x_0, y_0) votes for a node if $f_1 < 0$ and $f_3 > 0$. The first condition, $f_1 < 0$, yields

$$L(b_1 - y_0) + \ell x_0$$

or $$x_0 < \frac{-L(b_1 - y_0)}{\ell_1}$$

similarly, $f_3 > 0$ yields,

$$x_0 > \frac{-L(b_2 - y_0)}{\ell_2}$$

Therefore, for the region $0 \le a \le 1$

$$x_1 = -\frac{L(b_2 - y_0)}{\ell_2} \qquad (7)$$

and, $$x_2 = -\frac{L(b_1 - y_0)}{\ell_1} \qquad (8)$$

The above Eq. (7) and Eq. (8) are applicable for the first part of the slope. For the other parts their forms are given as follows:

Region $-1 \le a < 0$:

$$x_1 = \frac{L(b_1 - y_0)}{\ell_2} \qquad (9)$$

and, $$x_2 = \frac{L(b_2 - y_0)}{\ell_1} \qquad (10)$$

Region $1 < a < \infty$:

$$y_1 = -\frac{L(b_2 - x_0)}{\ell_2} \qquad (11)$$

$$y_2 = -\frac{L(b_1 - x_0)}{\ell_1} \qquad (12)$$

Region $-\infty < a < -1$:

$$y_1 = \frac{L(b_1 - x_0)}{\ell_2} \qquad (13)$$

and, $$y_2 = \frac{L(b_2 - x_0)}{\ell_1} \qquad (14)$$

The above procedure can be implemented efficiently if the image pixels are stored row wise starting from $h = 0$ to $h = H - 1$. The procedure is summarized as follows:

It is assumed that the two dimensional array $f[y, x], 0 \le y \le H - 1$ and $0 \le x \le W - 1$ stores the image. For an image pixel at (x_0, y_0), $f[y_0, x_0] = 1$ and $f[y_0, x_0] = 0$ for the background pixels. We use two one-dimensional arrays, $xedge_i$ and $yedge_i$, which

store x and y coordinates of image pixels. Also, np stores the total number of image pixels.

The voting process for the part $0 \le a \le 1$ is accomplished with the help of the following method.

Set $y_{old} = -1$.

Vote for node if $xedge_i \in [L \times \frac{(b_2 - yedge_i)}{l_2}, L \times \frac{(b_1 - yedge_i)}{l_1}], \forall i = 0,...np-1$.

Latest values of $L \times \frac{(b_2 - yedge_i)}{l_2}$ and $L \times \frac{(b_1 - yedge_i)}{l_1}$ are computed if $y_{old} \neq yedge_i$ and then y_{old} is replaced with $yedge_i$.

The same steps will be followed for the part $-1 \le a < 0$. The roles of x and y will be interchanged for the third and fourth parts.

The efficiency of the algorithm also depends on threshold, T. Its value is initially set to 1. As the solution progresses its value is updated at a leaf node if the votes received by the node is more than the current value of T. This facilitates the pruning of tree thereby reaching global maximum without exploring all nodes of a quadtree or binary tree.

A quadrant (or node) is represented by a rectangle $(a_1, b_1) \times (a_2, b_2)$ where initially, $a_1 = 0, a_2 = H, b_1 = -W, b_2 = W + H$. The maximum level of the tree is taken as $\lceil \log_2(2W + H) \rceil$ where $\lceil f \rceil$ defines the smallest integer greater or equal to f.

It is clear that the requirement of the storage is of the order: $\max_{edgepix} \times \max_{level}$ words. Where $\max_{edgepix}$ and $\max_{level}$ are maximum possible number of edge pixels and maximum levels that can be visited, respectively. If $\max_{edgepix} = 10^5$ and $\max_{level} = 16$, then the space requirement will be of the order $4 \times 16 \times 10^5 bytes \cong 6.4MB$, assuming that one word takes 4 bytes and these values are large enough to process an image of the size $1000 \times 1000 pixels^2$ assuming that only 10% of the total pixels constitute image pixels. The SHT requires $4 \times 3 \times 10^5 \times 180k\ bytes \cong 216MB$ for storing the votes in a 2-D integer array, $P[b, a]$ which does not include the storage for coordinates of image pixels on each line during the detection process whose order would be quite large.

Processing at the root node: We initialize the root node structure with following values: $a_1 = 0, b_1 = -H, a_2 = 180, b_2 = (W + H), votes = NE, child = -1$. Here, NE is the total number of edge pixels. Also, each edge pixel votes for the root node as, $level[0].vote\_in[i] = i, \forall i = 0,1,2,..., NE-1$. As we go finer with the resolution of Hough space, we keep on slicing the virtual Hough space into halves with respect to vertical and horizontal axes. Thus, $ac = \frac{a_1 + a_2}{2}, bc = \frac{b_1 + b_2}{2}$. This makes the structure of Hough space a quadtree. At every level, we choose North-West cell as child 0, North-East cell as child 1, South-East cell as child 2 and South-West cell as child 3 as described in Fig. 4. While we move finer with the resolution, the child node is tested against those image pixels only, which voted for the parent node. This is made possible by the use of $vote\_in$ member of the $node$.

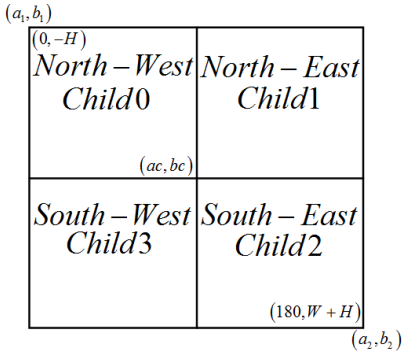

Fig. 4. Hough space quadtree subdivision process.

IV. EXPERIMENTAL RESULTS

A number of experiments are carried out to analyze performance of HHT over SHT with respect to speed and requirement of memory in skew detection in document images. All computations are carried out for a large number of binary documents which vary in size and script and are taken from different sources: books, newspapers, office documents and magazines. Results are presented only for three documents: *doc1*, *doc2* and *doc3*. The information of these document images are given in Fig. 4(a) through 6(c).

The first document which is in Roman script is small and its quality is poor. The second document has larger size which is a part of a bilingual document comprising of Roman and Gurumukhi scripts (Gurmukhi is a script used in Punjabi language, the 14th largest language in the world, predominantly used in India). The third document is in Roman and is comparatively quite large. All documents have skews of 5^0, 8^0, 15^0, 21^0, 30^0 and 44^0. We have restricted our analysis up to $\pm 45^0$ because skew more than $\pm 45^0$ rarely occurs in practical applications. Results for negative skew are not given because the results are identical. Skew in all these documents is introduced using Paint Shop Pro. The size of a document and number of image pixels in a document vary w.r.t. skew as is clear from Table 1.

Comparison of space, time and detected skew is made between SHT and MHHT in Table 1. The SHT requires a 2-D array, $P[3N][N]$, to accumulate votes, where $N = \max(H, W)$. It is assumed that the array is of the type integer and occupies four bytes of storage for each element. The order of memory requirement for MHHT is calculated using the expression, $s = 4N_b\ell$, where ℓ is the maximum height of the tree, because we need a one dimensional integer array, $vote\_in[s]$, at each level of the tree. The array $vote\_in[s]$ stores the list of points which currently belong to a node under consideration. Let $p = vote\_in[i]$, then the i-th point belonging to the node is the p-th point in the list of image points. This provides a major advantage over SHT as we can retrieve the *x*- and *y*- coordinates of points of the line having maximum votes. This feature also allows saving points on a line whenever we want to detect more than one line. The same feature can be incorporated in SHT but with higher cost due to a 3-D integer array $vote\_in[3N][N][MAXPOINTS]$, where $MAXPOINTS$ is the maximum number of points expected on a line. Clearly, the memory requirement would be considerably high. For example, if the size of an image is 512x512 pixel^2 and $MAXPOINTS$ is set to 512, then the memory required by the array is 4x3x512x512x512 bytes=1536 MB.

The detected skew angles by SHT and MHHT have almost the same behavior. The variation in detected skew by both algorithms from the actual skew in the first document is more than the variation observed in second and third documents. The larger variation is due to the poor quality of the first document. The speed of MHHT is much more than the speed of SHT which varies from a factor of 3 for larger documents to a factor of 14 for smaller documents. The variation in speed factor can be explained with the help of time complexity which is $O(N_bL)$ for SHT and $O(N_bN_n)$ for MHHT, where N_n is the number of nodes visited in MHHT. The growth in N_bL for SHT is much less as compared to the growth for N_bN_n for MHHT. This is clear for *doc1, doc2* and *doc3* at 5^0 skew where the value of $\frac{N_n}{L}(L=180)$ is 27.73, 85.22 and 626.14, respectively. This result reveals an interesting fact that MHHT is much faster for smaller documents which can be very effective in skew detection in document images as only a small part of the whole document is normally considered for this purpose [2]. Although the gain in speed for larger documents is less, MHHT still can be used because it has an added advantage of providing *x* and *y*-coordinates of points of detected lines with very less memory requirement. The requirement of memory by SHT grows rapidly with the size of image. On the other hand, memory requirement for MHHT is very less, the maximum size is 1.256 MB for *doc3* for MHHT while for SHT it is 27.834 MB, which is the minimum memory requirement for SHT when only voting is performed. When image points are also saved, the memory requirement for SHT is enormous and becomes unmanageable even on today's PCswhich come with very large memory.

V. CONCLUSION

A new algorithm for detecting skew in document images is proposed which is very efficient in memory and speed. The algorithm is based on recursively dividing the Hough parameter space and finding the global maximum using depth first traversal of a tree. Experimental results are presented to strengthen the concept.

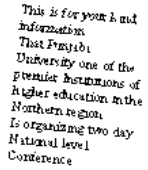
This is for your k nut
information
That Punjabi
University one of the
premier institutions of
higher education in the
Northern region
Is organizing two day
National level
Conference

Fig. 5(a):doc1

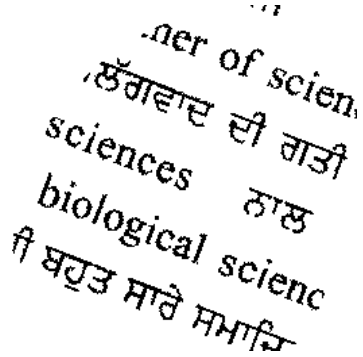

Fig. 5(b):doc2

Hough transform is one of the most frequently used operation in digital image processing since it has wide applications in Computer Vision, Pattern Recognition, Optical Character Recognition, Document Analysis, etc. Its main advantage is due to its robustness to noise even under degraded environment. In its conventional form it is easy to implement. However, the conventional form is limited to small size images because of heavy computational and memory requirements for large images, particularly engineering drawings [1]. As an example, size of an A0 size drawing is 13783 × 10078 binary pixels [1] which is difficult to be processed for line detection. Such images are processed either on large workstations or on mini and mainframe computers or special forms of Hough transform are employed to circumvent the problems. Fortunately modern PCs have large processing power and come with large memory (256 MB is normally the minimum memory available with PCs). Attempts are, therefore,

Fig. 5(c):doc3

REFERENCES

[1] L. O' Gorman, The document spectrum for page layout analysis, IEEE Trans. Pattern Analysis and Machine Intelligence, 15(11), 1993, 1162-1173.
[2] B. Yu, A. K. Jain, A robust and fast skew detection algorithm for generic documents, Pattern Recognition, 29(10), 1996, 1599-1629.
[3] H.F. Jiang, C.C. Han, K.C. Fan, A fast approach to the detection and correction of skew documents, Pattern Recognition Letters, 18, 1997, 675-686.
[4] E. Kavallieratou, N. Fakotakis, G. Kokkinakis, Skew angle estimation for printed and handwritten documents using the Wigner-Ville distribution, Image and Vision Computing, 20, 2002, 813-824.
[5] P. Shivakumara, G. H. Kumar, A novel boundary growing approach for accurate skew estimation of binary document images, Pattern Recognition Letters, 27, 2006, 791-801.
[6] A. V. N. Manjunath, G. H. Kumar, P. Shivakumara, Skew detection technique for binary document images based on Hough transform, International Journal of Information Technology, 3(3), 2006, 194-200.
[7] H. Li, M.A. Lavin, R.J.L. Master, Fast Hough transform: a hierarchical approach, Computer Vision, Graphics and Image Processing, 36, 1986, 139-161.
[8] J. Illignworth, J. Kittler, The adaptive Hough transform, IEEE Trans. Pattern Analysis and Machine Intelligence, 9(5), 1987, 690-697.
[9] J. Princen, J. Illignworth and J. Kittler, A Hierarchical approach to line extraction, Proc. IEEE Computer Vision and Pattern Recognition conference, San Diego, 1990
[10] N. Guil, J. Villalba, E.L. Zapata, A fast Hough transform for segment detection, IEEE Trans. Image Processing, 4(11), 1995, 1541-1548.
[11] Chandan Singh, Nitin Bhatia, and Amandeep Kaur, "Hough transform based fast skew detection and accurate skew correction methods", Pattern Recognition, Volume 41, Issue 12, December 2008, Pages 3528-3546.
[12] U. Pal and B. B. Chaudhuri, An improved document skew angle estimation technique, Pattern Recognition Letters, 17, 1996, 899-904.
[13] P. Y. Yin, Skew detection and block classification of printed documents, Image and Vision Computing, 19(8), 2001, 567-579.

Table 1. Comparison of memory requirement, execution time and detected skew for three document images skewed at various angles.

Document	size (pixel^2)	No. of image pixels, N_b	Skew (deg)	SHT			MHHT						
				Memory (MB)	CPU (sec)	Detected skew (deg)	Memory (MB)	CPU (sec)	Detected skew (deg)	Height of tree	Total nodes	Nodes visited, N_b	Percentage of nodes visited
doc1	128x128	1 764	5	0.196	0.453	5.35	0.056	0.031	5.99	8	21 845	4 993	22.85
	146x146	1 704	8	0.256	0.453	8.88	0.054	0.031	8.02	8	21 845	4 813	22.03
	161x161	1 697	15	0.311	0.453	17.35	0.054	0.031	15.37	8	21 845	4 385	20.07
	172x172	1 701	21	0.355	0.453	20.55	0.054	0.016	21.57	8	21 845	3 789	17.34
	185x185	1 706	30	0.41	0.453	30.7	0.054	0.016	29.71	8	21 845	3 625	16.59
	195x195	1 716	44	0.456	0.469	44.09	0.054	0.031	43.79	8	21 845	4 517	20.68
doc2	317x317	6 514	5	1.206	0.656	5.36	0.234	0.156	5.24	9	87 381	15 341	17.56
	307x307	6 518	8	1.131	0.641	8	0.234	0.141	8.09	9	87 381	14 049	16.08
	281x281	6 511	15	0.948	0.625	14.87	0.234	0.109	15.09	9	87 381	10 313	11.81
	256x256	6 511	21	0.786	0.61	21.33	0.234	0.125	21.23	9	87 381	11 853	13.56
	293x293	6 493	30	1.03	0.641	29.35	0.234	0.094	30.86	9	87 381	7 805	8.93
	336x336	6 528	44	1.355	0.671	44.09	0.234	0.11	44.39	9	87 381	6 725	7.7
doc3	1083x1083	29 221	5	14.074	7.562	4.91	1.256	2.64	5.33	11	13 98 101	1 12 705	8.06
	1138x1138	29 094	8	15.54	8.188	8	1.25	2.625	8.39	11	13 98 101	1 00 573	7.19
	1255x1255	29 094	15	18.9	9.265	15.08	1.25	3.03	15.04	11	13 98 101	1 05 157	7.52
	1340x1340	29 070	21	21.547	10.203	20.94	1.25	3.485	21.17	11	13 98 101	1 27 717	8.78
	1439x1439	29 100	30	24.849	11.328	30.03	1.252	2.344	30.11	11	13 98 101	63 745	4.56
	1523x1523	29 044	44	27.834	12.343	44.21	1.248	2.36	43.94	11	13 98 101	63 045	4.51