Pattern Analysis & Applications (2000)3:243–253

A Document Skew Detection Method Using the Hough Transform

A. Amin and S. Fischer

School of Computer Science and Engineering, University of New South Wales, Sydney, NSW, Australia

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 flat-bed 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 realising a practical document reader. In this paper we describe a new algorithm for skew detection. We then compare the performance and results of this skew detection algorithm to other published methods from O'Gorman, Hinds, Le, Baird, Postl and Akiyama. Finally, we discuss the theory of skew detection and the different approaches taken to solve the problem of skew in documents. The skew correction algorithm we propose has been shown to be extremely fast, with run times averaging under 0.25 CPU seconds to calculate the angle on a DEC 5000/20 workstation.

Keywords: Connected components; Document analysis; Hough transform; Least square method; Projection profile; Skew detection

1. INTRODUCTION

Over the past two decades, many people have attempted to solve the problem of skewed documents using several different methods. Applying the Hough Transform to a set of points to determine straight lines in the image has been a popular approach. Srihari and Govindaraju [1] use the Hough Transform on all black pixels, while Hinds et al [2] reduce the data with the use of horizontal and vertical run length computations. Le et al [3] apply this transform only to the bottom pixels of a connected component.

A theoretically identical method is the use of 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 maximised. Baird [4] uses connected component analysis and creates a projection profile using a single point to represent each connected component. Postl [5] briefly discusses a method which maximises a given function, called the premium, with respect to the angle by simulating scan lines. Ishitani [6] detects the skew angle by maximising the deviation from the mean of the pixel counts in the projection profiles. Akiyama and Hagita [7] describe a method that is extremely fast, where the document is segmented into vertical partitions, and projection profiles are created for each partition. The skew angle is then calculated by determining the average shift in zero crossings between partitions.

Hashizume et al [8] propose a method that computes the nearest neighbour of each character and creates a histogram of these values to detect the skew angle. Liu et al [9] 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. O'Gorman [10] discusses skew correction in the framework of his 'docstrum' analysis. Here, the skew angle is determined by finding the nearest neighbours for a connected component and determining the most common angle between neighbouring connected components. This method generally searches for the five nearest neighbours, which in a normal situation will correspond to the connected components on either side, above, and below the connected component in question, as well as one extra in case there is noise of some sort. O'Gorman's method can be seen as a generalisation of the previous methods, although the data

Received: 21 November 1998
Received in revised form: 25 August 1999
Accepted: 20 October 1999

is used to extract additional information about the image as well.

Two other methods are not so easily grouped. Postl [5] uses a two-dimensional Fourier transform, from which the premium function can then be calculated and maximised. Smith [11] discusses an algorithm that groups the connected components into lines, moving from left to right across the document, and uses a continually updated estimate of the skew angle to place components in the correct line.

The interested readers can refer elsewhere [12,13] for a comprehensive survey of different methods for document image skew detection.

2. NEW SKEW DETECTION ALGORITHM

2.1. Overview

The proposed skew detection algorithm attempts to 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. In each text block, we attempt to isolate the last line and determine its angle. We first use the Hough Transform to estimate the angle, and then obtain a more precise figure by applying the Least Squares formula 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.

As the experimental results later in the paper show, this method is accurate for documents skewed at angles up to 45°. At greater angles, the identified bottom line may in fact be one of the edges of the block of text. The other constraint of this method is that it fails in some cases where part of the document is cut off, because the calculated bottom line may be in fact simply the bottom-most characters that were scanned. Obviously, if there are many blocks on the page, then this constraint will not be a problem.

2.2. Preprocessing

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 minimised.

Since most of the algorithms in the experiment were designed for binary images, where each pixel is either black or white, a global threshold for the image must be chosen, separating the background from the data. We use a modified version of Otsu's thresholding method [14], where we create a histogram of all pixel values. After smoothing the histogram, it is inspected for local maxima. In a grey-scale image, there will generally be two peaks – one corresponding to the text, and one corresponding to the background. After finding these peaks, we apply Otsu's formulas to find an appropriate threshold point.

2.3. Connected Component Analysis

After the preprocessing is completed, the connected components must be determined. Connected components (CC) are bounded by rectangular boxes bounding together regions of connected black pixels. The objective of the connected component stage is to form rectangles around distinct components on the page, whether they be characters or images. These bounding rectangles then form the skeleton for all future analysis on the page.

The algorithm used to obtain the connected components is a simple iterative procedure which 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.

2.4. Grouping

After the CCs have been determined, the next step is to group neighbouring connected components of similar dimensions. All the CCs of documents fall into one of three categories: *noise*, *small* or *large*, depending on their size. The noise CCs are then removed from any further skew calculation, and the other CCs are then merged in accordance to the category they fall under. Merging requires the use of a prefixed threshold (different for each category) so as to provide a means of determining the neighbourhood of a group.

The grouping algorithm [15] 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 neighbourhood 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. As can be seen in Fig. 1, the small CC (hashed) is merged into the right group and the group dimensions increase to accommodate the new

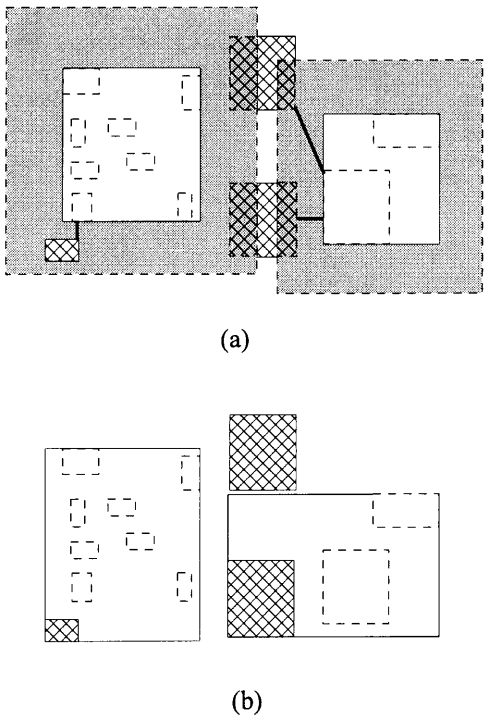


Fig. 1. Merging criteria of a CC into an existing group.

CC. Both larger CCs are near the group of large CCs, but only the lower one is close to an individual CC. Therefore, a new group is created for the upper CC, and the lower one is merged into that group.

As can be seen from Fig. 1(a), the neighbourhood of a group (represented by the grey shaded region surrounding each group) is the extent of the region of influence which the group possesses over a CC. In other words if the CC lies within the group's neighbourhood it is merged into the group (the dashed rectangles in the group represent previous CCs that have been successfully merged into the group). If the CC does not fall within the group's neighbourhood then it's not merged into the group. Furthermore, CCs from different categories are not merged together, thus as can be seen from Fig. 2(a) the large (hashed) CCs are not merged with the group to the left (even though they fall in its region of influence) since they are much larger than the CCs which comprise that group. Figure 2 gives an example of the grouping process.

2.5. Skew Estimation

To estimate the skew angle for each group, we divide each group into vertical segments of approximately the width of one connected component and store only the bottom rectangle in each segment. This process allows us to store primarily the bottom row of text in each group, with other random connected components where there are gaps in the bottom row. We then apply the Hough Transform to the point at the centre of each rectangle, mapping the point from the (x,y) domain to a curve in the (ρ,θ) domain according to Eq. (1):

$$\rho = \mathbf{x}\cos\theta + \mathbf{y}\sin\theta, \text{ for } 0 \leq \theta < \pi \tag{1}$$

The Hough Transform has several interesting properties:

1. Points in the (x,y) domain map to curves in the (ρ,θ) domain.
2. Points in the (ρ,θ) domain map to lines in the (x,y) domain, where ρ is the perpendicular distance of the line from the origin, and θ is the angle from the horizontal of the perpendicular line.
3. Curves that cross at a common point in the (ρ,θ) domain map to collinear points in the (x,y) domain [16].

When the Hough Transform has been applied to each point, the resultant graph is examined to find the point at which the most curves cross. This point will correspond to a line of connected components, which should represent the bottom row of text. The extra connected components that were mapped into the (ρ,θ) domain will not cross at the same point and are excluded from further calculation of the angle.

2.6. Skew Calculation

If more than one curve crosses at this point in the graph, we then determine the slope of a straight line that best approximates these points, which represent the connected components of the bottom row of text, using the least squares method.

Least squares is a statistical method for finding the equation (line, quadratic, etc.) of best fit given a set of points. We determine the equation of best fit for a line

$$y = a + bx \tag{2}$$

where the coefficients a and b are computed using the formulae [17]

$$b = \frac{n\sum_{i=1}^{n} x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \tag{3}$$

$$a = \frac{\sum_{i=1}^{n} y_i - b\sum_{i=1}^{n} x_i}{n} \tag{4}$$

given that (x_i, y_i) is a point from a set of samples $\{(x_i, y_i) \mid i = 1, 2, \ldots, n)\}$.

Analogously, the CCs associated with each line segment constitute the sample space. Once again these CCs are denoted as points, and noting that each line segment maintains a set of points, i.e. $\{(x_i, y_i) \mid i = 1, 2, \ldots, n)\}$, the above equations are applied to determine the equation of the line of best fit for a given line segment. Given that the equation derived is of the form given in Eq. (2), we can determine the angle α□ which the line makes with respect to the horizontal by using Eq. (5)

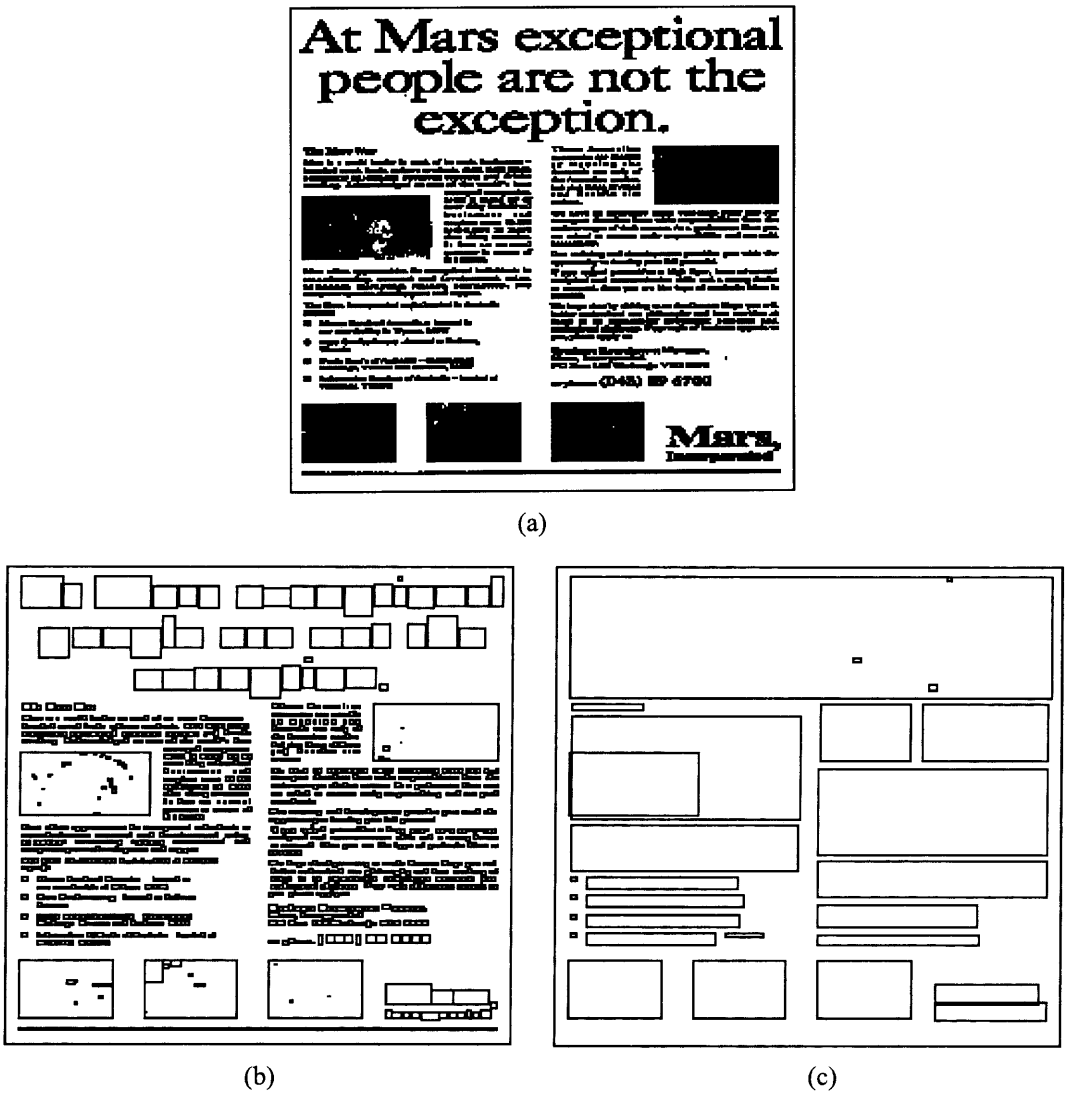


Fig. 2. Example of connected components and grouping (a) Original image, (b) connected components, (c) groups.

$$\alpha = \tan^{-1}(b) \tag{5}$$

where b represents the gradient of the line. The slope of the calculated line gives the dominant skew angle for the group. After we calculate an angle for each group, we group them in sets of 0.5°, weighting the angle by the number of connected components used in its calculation. We then determine which partition contains the most angles, and average these values to determine the skew angle, α.

2.7. Skew Correction

Once the skew angle has been determined, the image is then skew corrected to lie horizontally on the page. The method was inspired by Paeth [18]. To calculate the correct value for a pixel at location (x,y) in the skew corrected image, we determine the true original position (x_{old},y_{old}) of the pixel using the formulae in (6):

$$\begin{aligned} x_{old} &= x \cos \alpha + y \sin \alpha \\ y_{old} &= y \cos \alpha - x \sin \alpha \end{aligned} \tag{6}$$

where α is the calculated skew angle of the image.

Then, since (x_{old},y_{old}) is generally non-integral, a weighted average of the surrounding pixels is calculated to determine the value of (x,y). If (x_{old},y_{old}) lies outside the original image, the new pixel is given a value of white.

3. OTHER SKEW DETECTION ALGORITHMS

3.1. Baird's Algorithms

Baird's algorithm 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

tables, variable line spacing, mixed fonts, and a wide range of point sizes. Baird [19] 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. Although the noise and character fragments are the sources of skew detection error, Baird claims that they will not noticeably affecting the detection of skew angle.

The energy alignment measure function given in Eq. (7) has a global maximum at the correct skew angle. Baird comments that any positive superlinear function will provide the same result:

$$A(\theta) = \sum_{i=1}^{m} c_i^2\ (\theta) \tag{7}$$

For highest accuracy, he states that the skew angle should be limited to $\pm 15.0°$. The accuracy depends upon the angular resolution of the projection profile.

Baird states that computing this function for all angles at intervals can be time consuming. He therefore proposes an iterative sampling procedure to locate the global maximum. He states that around the maximum there always appears a roughly symmetrical peak, whose slopes are smooth with slowly rising bases and steep tops, with a sharp maximum. This shape can be closely approximated as the intersection of two functions S^+ and S^-, of the form given in Eq. (8):

$$S^{\pm}(\theta) = \frac{a}{b \pm c\theta} \tag{8}$$

where a, b and c are real parameters (different for S^+ and S^-).

The algorithm first locates the characteristic shape of the peak by searching at a course resolution of 0.7°. Then it fits the slopes with approximating functions of the form shown above using an iterative non-linear separable least-squares fitting algorithm. The point at which the approximations intersect is the first refined estimate of the peak. Samples are then calculated at finer resolution to refine the location.

3.2. Hinds et al Algorithm

Hinds et al [2] 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 $\pm 45°$. The accuracy depends upon the angular resolution of the Hough transform.

3.3. O'Gorman's Algorithm

O'Gorman [10] 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 neighbour clustering. The following steps take place:

1. Preprocessing is used to remove noise. O'Gorman's own method to reduce the salt and pepper noise is the k-fill filter. This removes 'small' (determined by chosen dimension of k pixels) regions on text edges ('spurs') or in background, and likewise, it fills in small holes. The scheme is iterative and each iteration consists of two subiterations doing on-fills and off-fills, respectively. Since the filter is quite slow and a completely clean document is not necessary for skew estimation, the filter process was not used in testing.
2. The next stage of preprocessing is to find the connected components (CC).
3. The fundamental stage of this method is to find the k nearest neighbours (NN) for each of the CC's (or more precisely, the bounding boxes) and to form a suitable data structure containing the results. A few preliminaries are used to speed up the process, such as sorting pointers to CC's in order of x-positions and cutting off the search when the x-distance exceeds the kth closest distance already found. The centroids of each box are used, and the five nearest neighbours are identified, as recommended by O'Gorman. The reason is that most commonly (except near the edges of the image or text areas), the four nearest neighbours are those immediately to the left, right, up and down directions (imagining that the image is suitably oriented), and a fifth NN is for even better coverage of the directed neighbours (such as between words).

   Now this data is plotted on a polar diagram using radial NN relative distance and NN angle ($k \times n$ points, where n is the number of CCs) and it is called the 'docstrum' by analogy with a power spectrum. Now the blob concentrations on the (almost symmetric) docstrum represent the inter-character spacing along the line and also less intense blobs along the same axis represent the inter-word spacing, while the blobs along the perpendicular axis represent the inter-line spacing.

   The integration of the docstrum over distance and angle produces two histograms. These are respectively, the NN angle histogram and the NN distance histogram. The NN angle histogram has a peak, and after suitable smoothing directly gives the first rough estimation of the

skew angle, at least modulo 90°. The actual orientation readily follows due to the text concentration properties which appear as two main peaks on the NN distance histogram, i.e. inter-character spacing is less than inter-line spacing and also the blob intensity is higher along the lines when compared to intensity across the lines. In fact, the NN distance histogram is separated into two histograms, given the rough skew, so that each has a single peak representing the two basic text spacings. Note that this method works for any skew angle, but it does not distinguish upside down text.

4. Using the rough skew, the CC data is rotated to approximately the correct orientation, and using the estimations of inter-character and inter-line spacing, the line segments of text are next constructed. Least squares lines are fitted to the centroids of the CCs making up the segments and a more accurate skew estimate is made as a correction to the rough skew.
5. Further construction gives the text lines from the line segments. Again, least squares lines are fitted to the centroids to give the final weighted estimate of the image skew angle.

3.4. Postl's Algorithms

Postl [5] discusses two methods for skew determination. Postl calls his first method the 'simulated skew scan' method. It amounts to maximising 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 optimisation 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, i.e. $[Z]^2$. Postl suggests that the zeroth moment of the power spectrum taken along the radius vector at the chosen slope angle, which becomes the sum of squares of the projection profile bin heights. Put another way, this cost function is equivalent to the variance of the projection profile bin heights as the second central moment is obtained by subtracting the mean squared, which is just a constant. This latter cost function, derived from the Fourier version suggested by Postl, corresponds to a cost function used by many other authors, although without the Fourier connection.

3.5. Le et al Algorithm

This algorithm [3] applies the connected component analysis to the original image. Large connected components, which are classified as images, or small connected components, which are classified as noise or character fragments, are excluded from the calculation in order to help minimise the skew angle error. Next, it extracts the pixels of the last black runs of each connected components. Finally, the Hough transform is applied to each of these extracted pixels and the skew angle is chosen from the Hough accumulator array by searching for maxima.

This method has been tested with a large image database which contains several thousands of images and claims an accuracy of about 0.50°.

3.6. Akiyama and Hagita's Algorithm

Akiyama and Hagita [7] divide the page into columns (assuming the text is horizontal) and then calculate horizontal projection profiles for each column. Each peak in the profile corresponds to text line in the column. The skew angle α is obtained by calculating the arctangent of the phase shift β between adjoining projection profiles. This method has been shown to be extremely fast, but requires extremely regular text and cannot handle graphics. In addition, the algorithm cannot calculate skew angles of more than about 5° with any sort of accuracy, because the projection profiles will overlap too far and no accurate determination will be possible.

3.7. Discussion

Several factors influence the speed and accuracy of a skew detection algorithm, and the goals of a fast algorithm and one which is highly accurate tend to be conflicting. Basically, the more data that is used in the calculations, the better the result will be, but the longer the process will take. Therefore, the amount of information used is extremely important. Another factor affecting accuracy is what data is used in the calculation. For the result to be accurate, the effect of noise must be minimised, so that calculations are performed on meaningful values. Similarly, any images in the scanned page must be somehow removed from the data, either by identifying them beforehand with a separate algorithm, removing them in the process of calculating the skew angle, or by reducing them to a very small number of data points and assuming that they will be greatly outnumbered by the data points representing the text. The last factor is the actual calculations. If each data point is processed once and a final array is searched for maxima, as in the Hough Transform, then the algorithm will perform better than one which requires extensive searching.

3.8. Amount of Data

The amount of data used can range from every single pixel, such as in Srihari's [1] Hough transform method or Postl's

[5] algorithms, to a single point representing each connected component, as in Baird's [4] methods, among others. Data reduction, in addition to reducing run time, can assist to reduce the effect of noise. For instance, applying the Hough transform to a very noisy image may not give good results, but removing small connected components effectively cleans the image. Both Le [3] and Hinds [2] look at a subset of pixels of each connected component, which they assert to be the most important bits of information. Our method, as well as Baird [4] and O'Gorman [10], determines a single point which defines each connected component and performs the calculations on these points. These methods obviously run faster because the amount of data is three orders of magnitude less than the amount used in a method which performs calculations on every point in the image. Akiyama's [7] method uses the least data because the calculations are performed on a small number of projection profiles.

3.9. Type of Data

Individual data points can represent pixels, connected components, or some intermediate step between the two. O'Gorman [10] uses all connected components to determine the skew angle. He does, however, assume that the image contains only text and no images. Baird [4] uses all connected components which are smaller than a certain size, so noise is included in the calculations but images are excluded. Our method excludes both large and small connected components, attempting to isolate the text from the noise and images. Although these methods use the fewest data points, they obtain comparable results because text in a given line is generally similar in size, so these points should also be collinear. These algorithms also reduce noise as discussed above, and images are reduced to very few points, since images are almost always identified as a single connected component. All these algorithms use the assumption that identifiable lines can be extracted from this data, and that the noise and images are largely excluded. If noise connects characters from adjoining lines, or if the noise forms large enough connected components to be identified as data, then these methods lose some accuracy.

Le [3] identifies the bottom pixels of each connected component and uses this data in his calculation. Hinds [2] uses a similar approach, but he weights each bottom pixel with the number of pixels in that connected component that lie in the same column and are identified as part of the data. These two methods make the reasonable assumption that good lines that approximate the skew angle can be extracted from the bases of the connected components. Le [3] removes noise by a size analysis of the connected components, while Hinds [2] reduces the effect of noise by the pixel weighting, which gives more weight to larger connected components. Since these methods are so similar, one would expect their test results to be close, and Table 1 shows that the average run time for the two methods is almost identical.

Akiyama [7] calculates a series of projection profiles which correspond to columns on the page. Their claim is that the skew angle can be determined by inspecting the phase shift of these projection profiles, which can only be valid if the document consists solely of text of reasonably uniform font and which has lines that span the width of the page. Random noise will not affect the calculations, but coherent noise could affect the results.

Srihari [1] assumes that the coherency of the data will make the peaks in the Hough transform array easy to identify. This assumption will hold true if the noise is randomly distributed and should produce very accurate results. This method would only fail if the noise in the image has a strong linear character (for instance, if a scanner or photocopier produces streaks), or if there are images in the text which have identifiable lines.

Postl [5] calculates the skew angle directly by applying the Fourier Transform to the entire image and examining a half plane of the power spectrum coefficients. Random noise will not affect this method, since the Fourier Transform tends to minimise the effect of noise with respect to the data.

Table 1. Experimental results

	Average CPU seconds	Average error	Percentage of tests within error tolerance		
			<0.25 degrees	<0.5 degrees	<1 degree
New (Least Square Method)	0.24	0.19	74.29	85.71	97.14
Baird	0.11	0.27	59.46	81.08	97.30
Baird-2	0.13	1.16	18.52	29.63	62.96
Le	0.82	0.33	94.29	97.14	97.14
Hinds	0.84	0.52	78.79	84.85	90.91
Postl	2.23	0.33	46.88	90.63	93.75
O'Gorman	0.46	1.35	33.33	46.67	60.00
Akiyama	0.06	4.34	10.81	16.22	27.03

3.10. Calculations

Most of the algorithms discussed perform simple calculations on the data and then maximise some function to determine the skew angle. Our proposed algorithm is slower because the connected components must be grouped by size and proximity. This step can take a large amount of time if there are many connected components. However, this step is an important component of the algorithm because it attempts to identify separate blocks of text, which in turn improves the accuracy. Baird's [4] second algorithm uses an iterative procedure to locate the maximum of his energy alignment function, but there is little or no loss in speed because the first calculations are only performed at very coarse angles, and more precise angles are calculated once the search has been narrowed.

4. EXPERIMENTAL RESULTS

Over 120 document images were used in the experiments. These documents were unconstrained because they were chosen from a wide range of sources including technical reports, Arabic, Chinese and English magazines and business documents, business cards, Australian Telecom Yellow pages and badly degraded photocopies. These images contain a wide variety of layouts, including sparse textual regions, dense textual regions, mixed fonts, multiple columns, tabular and even for documents with very high graphical contents. Figure 3 shows two sample test images.

In addition to our method, we tested the above algorithms by selecting 40 from 120 plus unconstrained images, checking for both speed and accuracy, running each algorithm 10 times on each image.

The experiment is performed by rotating the original unskewed image at several known angles up to 45° and then running each method to calculate the skew angle. The skew detection error is the difference between the rotation angle and the detected skew angle. The time taken is measured from the end of any preprocessing required, such as thresholding, to the completion of skew angle detection. To obtain a more accurate measurement of the processing time, we ran the test 10 times on each image for each algorithm. The minimum of these 10 periods was determined and used to calculate the average time taken in the skew detection process.

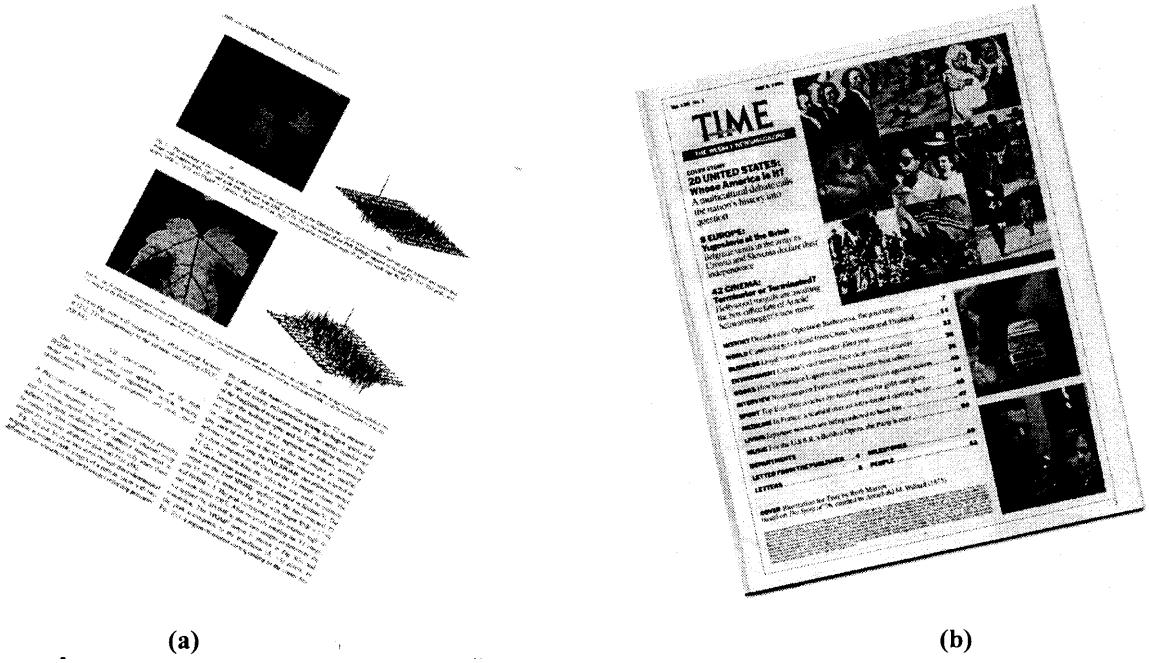
A given algorithm was only tested on angles up to its stated detection angles, and algorithms such as O'Gorman's docstrum method were only tested on text images if it was given as a restriction.

When testing the algorithms, we compared several factors: CPU speed, the average absolute skew angle error and the percentage of images whose skew was correctly detected within 0.25°. The figure of 0.25° was chosen because in a document with 8 point text, the bounding boxes of lines of text will begin to overlap at about this angle. If the document is skewed less than 0.25°, document analysis methods will almost certainly function properly. In addition, we looked at the percentage of images whose skew was correctly detected within 0.5° and 1°.

Table 1 shows the relative accuracy and speed for each algorithm. The first column shows the average speed in CPU seconds to calculate the skew angle. The second column shows the average absolute deviation from the theoretical skew angle. The last three columns show the percentage of images for which the skew angle was correctly determined within 0.25°, 0.5°, and 1.0°, respectively.

Clearly, Akiyama's algorithm [7] is faster than the other methods, with Baird's [4] methods and our method being next. When comparing accuracy, our algorithm returns the smallest average error of any tested algorithm. On the other hand, Le's algorithm [3] calculated almost 95% of the skew angles within 0.25°, while ours and Hinds' [2] methods calculated 75% and 78%, respectively. However, Baird's algorithm [4] does not perform as well in this area. O'Gorman's method [10] did very poorly in computing the skew angle, possibly because of the inaccuracy in calculating the nearest neighbours in the line, and Akiyama's method [7] returned very poor results, presumedly because the data used in the test was skewed at too high an angle for their algorithm or contained too much noise.

Figure 4 shows a runtime comparison of the algorithms. It can be seen that Postl's method [5] took a long time to determine the skew angle. This anomaly is because the



(a) (b)

Fig. 3. Two sample images. (a) Skewed at 27.5°, (b) Skewed at −12.3°.

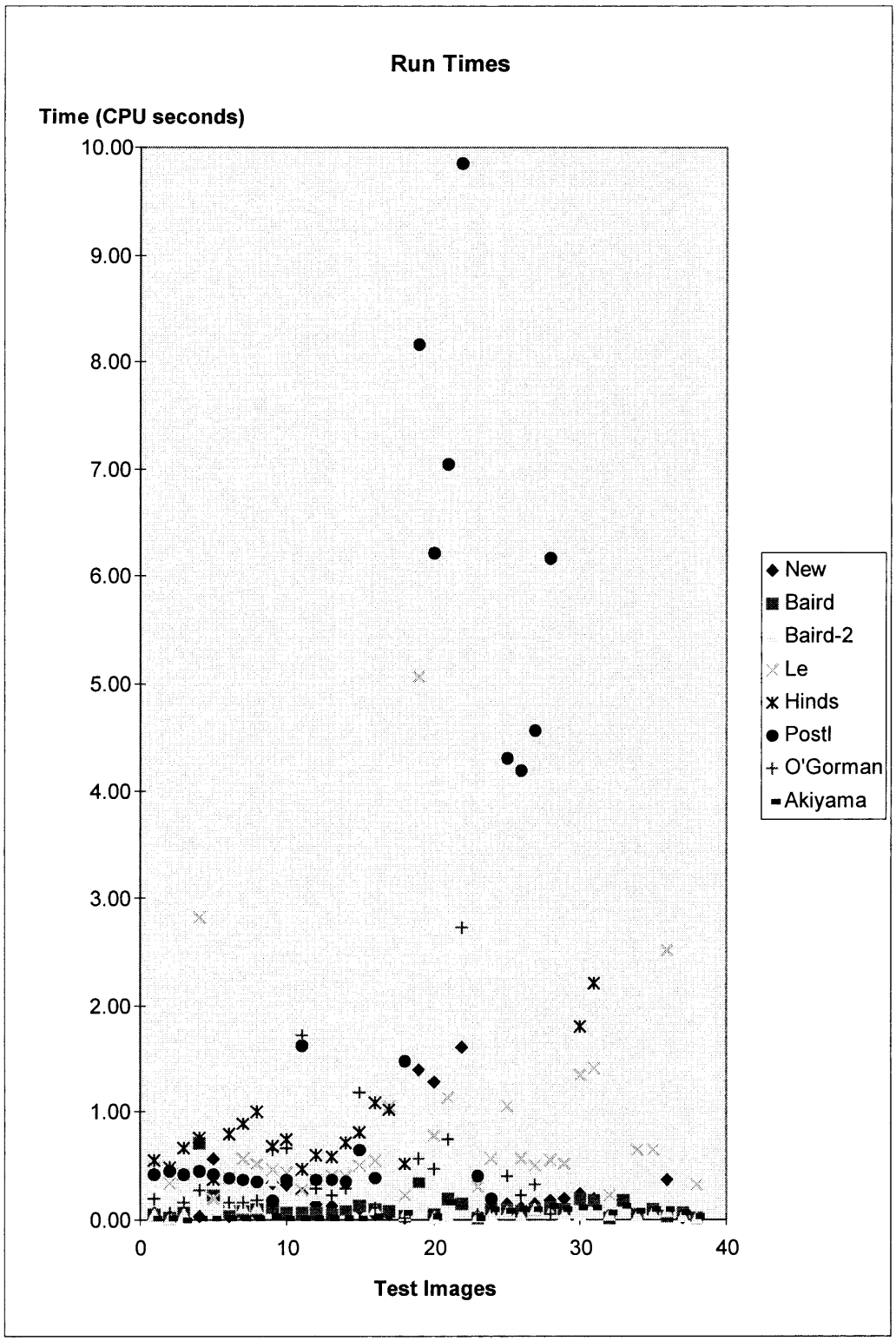
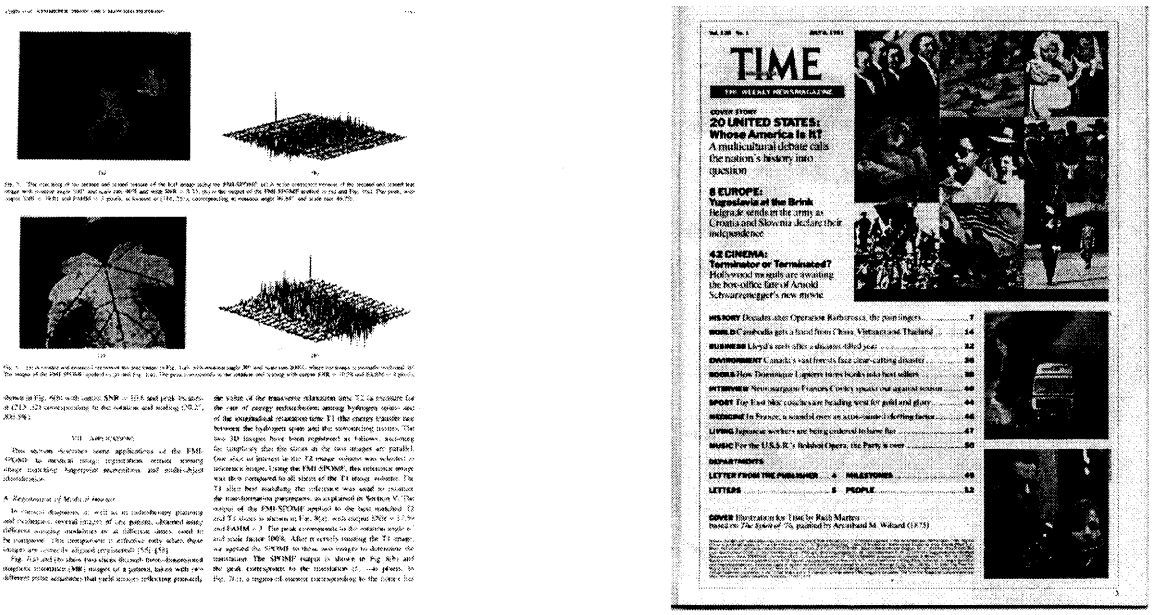


Fig. 4. Run-time comparison.



(a) (b)

Fig. 5. Skew corrected images of Fig. 3.

method is highly dependent on a proper threshold calculation. If the threshold is set high enough that some noise is included in the image, the Fourier Transform takes much longer to run. Other methods are less affected because the noise is filtered out before skew calculation, primarily by connected component analysis, or simply ignored.

In addition, we have implemented Srihari's method [1] using the Hough Transform. Since this algorithm applies the Hough Transform to each pixel, we found it prohibitively slow to test. Furthermore, the algorithm is limited to text images only. However, preliminary tests show that the results are very accurate.

Figure 3 shows two sample images from the test database. Figure 4(a) is a journal article that contains images and line drawings as well as text. Figure 4(b) is the table of contents from a magazine, which has multiple fonts and text sizes, pictures and graphics. Figures 5(a) and 5(b) illustrate the two images after skew correction. This range of images comprise the test images used in this experiment.

5. CONCLUSION

This paper has presented a new skew detection algorithm based on Hough transform In addition, we have implemented other well-known published algorithms and compared them for speed and accuracy. To-date, there has not been a method that is both more accurate and faster than our algorithm. There is a fundamental trade-off between speed and accuracy in any such algorithm, and we believe that our method manages to balance the two criteria well. Since we use the Hough Transform, we can detect large skew angles in a document, and tests show that the method is accurate for angles of up to 45°.

References

1. Srihari SN, Govindaraju, V. Analysis of textual images using the Hough Transform. Machine Vision and Applications 1989; 2:141–153
2. 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
3. Le DS, Thoma GR, Wechsler H. Automated page orientation and skew angle detection for binary document images. Pattern Recognition 1994; 27(10):1325–1344
4. Baird HS. The skew angle of printed documents. Proceedings of Society of Photographic Scientists and Engineers 1987; 40: 21–24
5. Postl W. Detection of linear oblique structures and skew scan in digitized documents. Proceedings 8th International Conference on Pattern Recognition 1986: 687–689
6. Ishitani, Y. Document skew detection based on local region complexity. IEEE 1993; 7:49–52
7. Akiyama T, Hagita N. Automated entry system for printed documents. Pattern Recognition 1990; 23(2):1141–1154
8. Hashizume A, Yeh P-S, Rosenfeld A. A method of detecting the orientation of aligned components. Pattern Recognition Letters 1986; 4:125–132
9. 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
10. O'Gorman L. The document spectrum for page layout analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 1993; 15(11):1162–1173
11. Smith R. A simple and efficient skew detection algorithm via text row accumulation. Proceedings 3rd International Conference on Document Analysis and Recognition 1995; 2:1145–1148
12. Hull JJ. Document image skew detection: Survey and annotated Bibliography. In: Document Analysis Systems II, JJ Hull, SL Taylor, Eds. World Scientific, 1998: 40–64
13. Spitz AL. Analysis of compressed document images for dominant

skew, multiple skew, and logotype detection. Computer Vision and Image Understanding 1998; 70(3):321–334

14. Otsu N. A threshold selection method from grey-level histograms. IEEE Transactions on Systems, Man, and Cybernetics 1979; 1:62–66
15. Amin A, Fischer S. Fast algorithm for skew detection. IS&T/SPIE Conference on Real-Time Imaging, USA, 1996; 65–76
16. Duda R, Hart P. Use of the Hough Transformation to detect lines and curves in pictures. Communications of the ACM 1972; 15(1):11–15
17. Walpole RE, Myres RH. Probability and Statistics for Engineers and Scientists, 4th ed. Macmillan, 1990: 362
18. Paeth A. A fast algorithm for general raster rotation. Proceedings Graphics Interface Vision Interface 1986: 77–81
19. Baird HS. Anatomy of a versatile page reader. Proceedings of the IEEE, 1992; 80(7):1059–1065

Adnan Amin received a BSc degree in Mathematics and a Diploma in Computer Science from Baghdad University in 1970 and 1973, respectively. He received a DEA in Electronics from the University of Paris XI (Orsay) in 1978, and presented his Doctorate D'Etat (DSc) in Computer Science to the University of Nancy I (CRIN), France, in 1985. From 1981 to 1985, Dr Amin was Maitre Assistant in the University of Nancy II. Between 1985 and 1987 he worked in INTEGRO (Paris) as Head of Pattern Recognition Department. From 1987 to 1990 he was an Assistant Professor at Kuwait University, and joined the School of Computer Science and Engineering at the University of New South Wales, Australia, in 1991 as a Senior Lecturer. Dr Amin's research interests are pattern recognition and artificial intelligence (document image understanding ranging from character/word recognition to integrating visual and linguistic information in document composition), neural networks, machine learning, and knowledge acquisition. He has more than 70 technical papers and reports in these areas. He is an associate Editor of *Pattern Recognition and Artificial Intelligence*, a member of the Editorial Board of *Pattern Analysis and Applications*, and has served on the program and technical committees of several international conferences and workshops. He is serving as Co-chairman of the *Second International Workshop on Statistical Techniques in Pattern Recognition* (SPR'98) and *Seventh International Workshop on Structural and Syntactical Pattern Recognition (SSPR'98)*, which were held in Sydney 1998. Currently, he is a chairman of Structural and Syntactical Pattern recognition, Technical Committee (TC2) of the International Association of Pattern recognition, Technical Committee (TC2) of the International Association of Pattern Recognition (IAPR). Dr Amin is a member of IEEE, the IEEE Computer Society, Australian Pattern Recognition Society and the ACM.

Correspondence and offprint requests to: A. Amin, School of Computer Science and Engineering, University of New South Wales, Sydney, NSW 2052, Australia. Email: amin@cse.unsw.edu.au