A Simple and Efficient Skew Detection Algorithm via Text Row Accumulation

Ray Smith

Hewlett Packard Laboratories, Filton Road, Stoke Gifford, BRISTOL, BS12 6QZ, UK.
rws@hplb.hpl.hp.com

Abstract

An important part of any document recognition system is detection of skew in the image of a page. This paper presents a new, accurate and robust skew detection algorithm based on a method for finding rows of text in page images.

Results of a comparison of the new algorithm against Baird's well-known algorithm on 400 pages show the new algorithm to be more accurate, robust and somewhat faster. In particular, the new algorithm only breaks down at skew angles in excess of 15 degrees, compared to the almost uniform distribution of breakdowns of Baird's algorithm.

1 Introduction

An important part of any document recognition system is detection and correction of skew in the image of a page. Page layout analysis and preprocessing operations used for character recognition depend on an upright image or, at least, knowledge of the angle of skew.[1][3][5][9] One example of a process which is spoilt by skew is the use of horizontal and vertical projection profiles. Projection profiles have many applications in document image processing and they rely on horizontal and vertical lines being aligned to the axes.

This paper presents a new, accurate skew detection algorithm based on a simple and robust method for finding rows of text independently of the skew angle of the image. By finding the rows of text, an accurate estimate of the skew angle of each text line, and thereby of the whole page, is obtained to a high degree of accuracy.

2 Previous work

A common method of skew detection is to simplify the Hough transform[4] by applying it to only a subset of pixels in the image.

The basic concept behind methods based on the Hough transform is the same:

1. Select some subset of pixels of the image which are few in number and most likely to form straight lines parallel to the baselines of the text rows.
2. For as many different directions as is necessary to achieve the desired accuracy, project the points selected in step 1 parallel to each direction in turn.
3. The direction for which the highest spikes are achieved in the projection gives the angle of the page skew.

Baird[2] and Nakano et. al.[5] use very similar methods by projecting the bottom of the bounding box of connected components parallel to a set of angles and choosing the angle which maximizes a measure of the variance of the counts in the projection.

Hinds et. al.[3] use vertical run-length coding and project the bottoms of the runs and Spitz[9] achieves a similar effect using Group 4 fax coding. Both of these methods avoid doing the connected component analysis required by Baird.

The basic idea behind the algorithm described in this paper was first outlined by Pratt et. al.[6], but has been improved significantly to cope with a much wider range of types of page.

3 Finding text lines independently of skew

The new algorithm was initially developed to solve the problem of finding the rows of text on a page in the presence of broken and joined characters, speckle noise and page skew, even in excess of 20 degrees.

The algorithm operates as follows:

1. Connected component analysis.
   Perform a connected component analysis[7] of the image. The connected components are (most likely) needed for the recognition processes anyway. For brevity, the connected components will be referred to as blobs. All references to size and location of a blob relate to the coordinates of its upright bounding box.
2. Filter blobs.
   The purpose of this process is to select a subset of the blobs which have a good chance of representing the

0-8186-7128-9/95 $4.00

body text. Exactness is unimportant – the key is in filtering out drop-caps, underlines and isolated noise. The blobs left out at this stage will be put in their place in the text lines later.

Blobs smaller than a fixed height are removed. The remaining blobs are then filtered by size. Blobs with height between the 20th and 95th percentile are retained, provided their width is not excessive.

3. Sort blobs.
   Sort the blobs (into ascending order) using x-coordinate (of the left edge) as the sort key. This sort makes it possible to track the skew across the page.
4. Make initial rows.
   Set the running average y shift to zero.
   For each blob in sorted order:
   Find the existing row which has most vertical overlap with the blob
   If there is no overlapping row
   Then
   Make a new row and put the blob in it.
   Record the top and bottom coordinates of the blob as the top and bottom of the row.
   Else
   Add the blob to the row.
   Expand the top and bottom limits of the row with the top and bottom of the blob, clipping the row height to a limit.
   Update the running average y shift with the bottom of the blob.
   Endif
   Endfor

   The sort ensures that blobs are processed in an order which gives some degree of overlap between adjacent blobs on a line. The sort also enables the use of a running average to measure the gradual vertical shift across the page. The running average y shift is used to vertically shift blobs when measuring their overlap with the rows. It is updated for each blob using:

$$Shift_{n+1} = \alpha Shift_n + (1-\alpha)\, Newshift$$

   where $0.5 < \alpha < 0.7$ and increasing to the larger end with an increasing number of rows on the page. The running average ensures that it is possible to track even large skew angles while being immune to descenders. Where there is a significant horizontal gap in the available blobs, the y shift is extrapolated across the gap.

   As the rows are accumulated, the top and bottom limits of the row are expanded to fit the inverse shifted y limits of the new blob. This helps the rows collect all the blobs which belong to them, rather than starting an extra row because the first blob on a line was small. The size of each row is restricted to a limit proportional to the upper quartile of heights from the filtering process. This prevents a row from expanding due to the different y positions of blobs on a skewed page, and growing so much that it collects blobs from more than one row.
5. Fit baselines.
   At the end of the above process, most of the blobs on the page are associated with some row. The arrangement is already good enough to obtain an accurate baseline for most of the rows on the page. This is done using a least median of squares fit.[8] The least median of squares fit is ideal for this application, as the baselines are close to straight, and there are outliers, (descenders, punctuation and other special characters) which would cause a least mean squares fit to exhibit significant error.

   The process of finding the global page skew stops here. The median gradient of the baselines provides a highly accurate estimate for the page.

4 Testing and results

The algorithm was tested by running it on a database of 400 A-size and A4 page images scanned at 300 dpi. These images are a mixture of real office documents varying in quality from original business letters and magazine pages to badly degraded photocopies and faxes. A significant proportion of the pages contain tables, pictures or figures and it is important that the skew detection is not distracted by the presence of these components.

The algorithm was tested against that of Baird,[2] since it is one of the most widely referenced skew estimation algorithms and Baird has also claimed it to be one of the fastest and most accurate reported.[1]

Two versions of Baird's algorithm were implemented and run on the same images. One version (Simple_Baird) applies a simple search of every angle in [-45,45] degrees with 10000 steps of 0.009 degree. The second version (Fast_Baird) applies a hierarchical search by an initial coarse search of every angle in [-45,45] degrees with steps of 0.9 degrees. This coarse search is improved by choosing the best angle and narrowing the search by a factor of 10 with a factor of 10 finer resolution. A third iteration is used to give a final effective search equivalent to Simple_Baird. The simple implementation gives an upper bound for accuracy and robustness, while the fast version gives an upper bound on the speed of Baird's algorithm.

To comprehensively test the accuracy of the algorithms, each was run on the original image and three rotations of the original, with the rotations chosen randomly (but the same for each algorithm) from (-28.6,28.6) degrees. The images were rotated in greyscale by bilinear interpolation[7] and then thresholded. Artifacts caused by the rotation were therefore negligible compared to the effects of thresholding.

The difference between the skew observed on the rotated image and the skew observed on the original should equal the rotation applied to the original. The scatter diagram in Figure 1 shows the change in skew angle measured by the new algorithm against the random skew applied for the 3 rotations of the 400 pages. It can be seen that for angles under 15 degrees, the new algorithm has no failures. Failures are mainly caused by the combination of a large rotation and a large image on the page. Images are often placed at the bottom right corner of a page where they

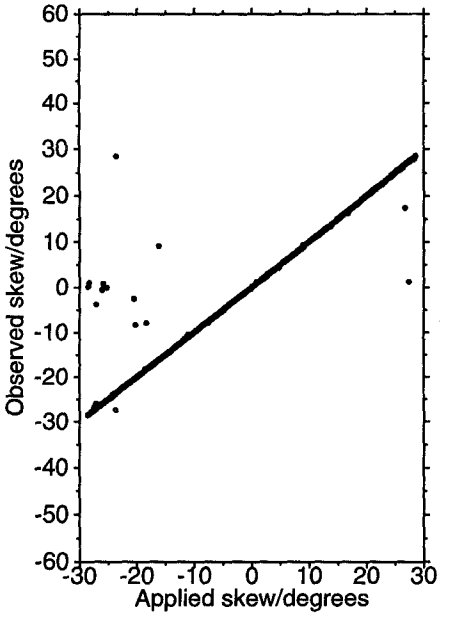

Figure 1 Scatter diagram of change in observed skew against applied skew for the new algorithm.

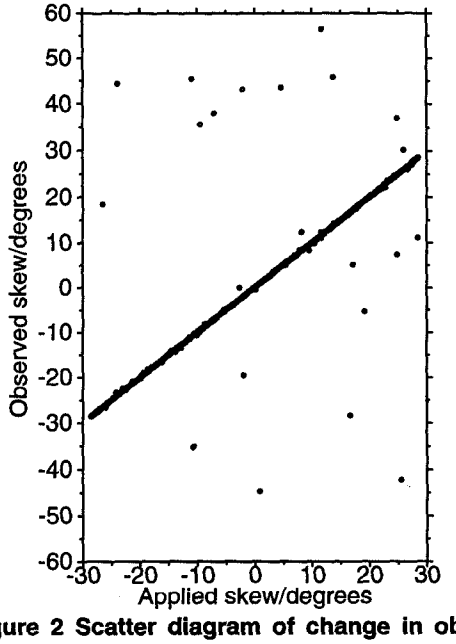

Figure 2 Scatter diagram of change in observed skew against applied skew for Simple_Baird.

offer greater interference with negative (clockwise) rotation – hence more failures with negative rotations.

Figure 2 shows the result of applying Simple_Baird to the same images. It is clear that both algorithms basically measure the skew with a high degree of accuracy, but that Simple_Baird has a greater tendency to fail dramatically, even on documents with almost no skew. It can also be seen from the $y=x$ line formed by the majority of the points that Simple_Baird also has a greater level of noise than the new algorithm.

Investigation of the failures of Simple_Baird showed them to be caused mainly by noise from figures/pictures and the background. Straight, inclined objects in figures or non-horizontal bands of noise can generate bigger spikes in the projection than the text and Baird's algorithm thus chooses the incorrect angle. This might be fixable by using a blob filter similar to the new algorithm.

The scatter diagram for the Fast_Baird algorithm is shown in Figure 3. As might be expected, the price paid for the higher speed is an increasing tendency to fail with a very large error. These additional failures are due to a local maximum in the variance of the projection profile at an incorrect angle, with the global maximum falling between angles in the coarse search.

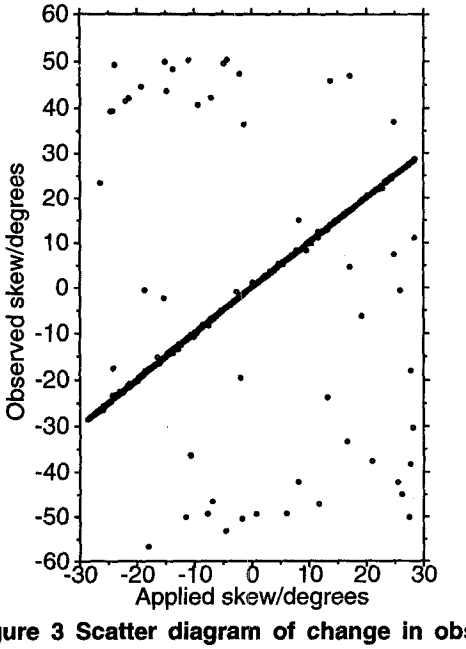

Figure 3 Scatter diagram of change in observed skew against applied skew for Fast_Baird.

The accuracy and time performance of the algorithms are shown in Table 1. The mean angle error really only confirms that there is no significant constant offset in the error of any of the algorithms. The Root-Mean-Square (RMS) angle error is the most statistically meaningful measure of angular accuracy. A more useful measure however, is in terms of vertical pixel error. The RMS pixel

error shows the vertical pixel error resulting from applying the angular error to the longest text line on the page.

Table 1. Accuracy and time performance of the skew detection algorithms

Algorithm	Mean angle err/ degrees	RMS angle err/ degs	RMS pixel err/ pixels	No. of out-liers	Mean CPU/ sec	SD CPU/ sec
CC analysis					10.11	9.44
New	0.216	2.83	77.8	15	0.32	0.28
Simp_Baird	-0.148	7.69	217	27	47.20	47.17
Fast_Baird	-0.284	11.7	341	57	1.02	0.94

Table 2. Accuracy of the skew detection algorithms without outliers

Algorithm	Mean angle err/degrees	RMS angle err/degrees	RMS pixel err/pixels
New	-0.0034	0.0718	1.87
Simple_Baird	0.0023	0.139	3.69
Fast_Baird	0.0061	0.132	3.86

Table 3. Accuracy of the skew detection algorithms using only skew angles of less than 15 degrees

Algorithm	Mean angle err/degrees	RMS angle err/degrees	RMS pixel err/pixels
Including outliers			
New	-0.0049	0.0583	1.26
Simple_Baird	0.150	6.64	163
Fast_Baird	0.124	10.1	226
Excluding outliers			
New	-0.0057	0.0529	1.27
Simple_Baird	0.0020	0.165	4.30
Fast_Baird	0.0031	0.147	4.19

The error figures are rather large, since they include all the results. Each algorithm failed drastically on some documents. Documents with pixel errors in excess of 100 are shown as outliers in Table 1.

The CPU timings relate to the Hewlett Packard 9000/ 720 workstation on which the algorithms were run. The line labelled "CC analysis" is the connected component analysis used by both Baird's algorithm and the new algorithm.

To obtain a better estimate of how the algorithms perform in cases where they do not fail drastically, the union of all the outliers was deleted from the statistics. The number of samples deleted was 73. The statistics resulting from the remaining common subset are shown in Table 2. These error figures are much more reasonable and show the new algorithm to have around half the error rate of Baird's algorithm.

Results for skew angles of less than 15 degrees are shown in Table 3. Results with and without outliers are given. As before, the union of all examples with a pixel error exceeding 100 are removed, being 27 in total. The new algorithm has no outliers under 15 degrees and thus its figures change only slightly with the removal of the outliers. Comparison of the figures with and without the outliers shows the significant benefit of the new algorithm in terms of its robustness for reasonable angles.

5 Conclusions

An approach to skew detection has been described which is quite different to that usually employed. The method accumulates rows of characters independently of skew and then computes the skew from the rows of characters. Using a large test set, the new method has been shown to be more accurate and somewhat faster than Baird's algorithm. Perhaps the most important feature of the new algorithm is that it has greater robustness with respect to noise, causing it to produce fewer wild estimates of skew, with no failures for angles under 15 degrees.

6 Acknowledgements

The author would like to thank Chris Newton and Phil Cheatle for their useful comments.

7 References

[1] Baird H.S. Anatomy of a versatile page reader. *Proc. of the IEEE,* **vol. 80** no. 7, pp1059-1065 July 1992.

[2] Baird H.S. The skew angle of printed documents. *Document Image Analysis,* L. O'Gorman, R. Kasturi, pp204-208. IEEE 1995.

[3] Hinds S.C, Fisher J.L, D'Amato D.P. A document skew detection method using run-length encoding and the Hough transform. *Proc. 10th Int. Conf. on Pattern Recognition,* Atlantic City NJ 16-21 June 1990 **vol. I,** pp464-468.

[4] Hough, P. Method and means for recognizing complex pictures, U.S. Patent no. 3069654. 1962

[5] Nakano Y, Shima Y, Fujisawa H, Higashino J, Fujinawa M. An algorithm for the skew normalization of document image. *Proc. 10th Int. Conf. on Pattern Recognition,* Atlantic City NJ 16-21 June 1990 **vol. II,** pp8-13.

[6] Pratt W.K, Capitant P. J, Chen W, Hamilton E.R, Wallis R.H. Combined symbol matching facsimile data compression system. *Proc. of the IEEE,* **vol. 68** no. 7, pp786-796 July 1980.

[7] Rosenfeld A, Kak A.C. *Digital picture Processing,* **vol. 2,** 2nd ed. Academic Press 1982.

[8] Roth G, Levine M.D. Segmentation of geometric signals using robust fitting. *Proc. 10th Int. Conf. on Pattern Recognition,* Atlantic City NJ 16-21 June 1990 **vol. I,** pp826-831.

[9] Spitz A.L. Skew determination in CCITT Group 4 compressed document images. *Proc. [First] Symposium on Document Analysis and Information Retrieval.* Tropicana Hotel, Las Vegas Mar 16-18 1992 pp11-25.