A New Method for Text-Line Segmentation for Warped Documents

Conference Paper · June 2010

DOI: 10.1007/978-3-642-13775-4_40 · Source: DBLP

CITATIONS
12

READS
326

5 authors, including:

Daniel Marques de Oliveira

12 PUBLICATIONS **65** CITATIONS

SEE PROFILE

Rafael Dueire Lins

Federal University of Pernambuco

263 PUBLICATIONS **2,862** CITATIONS

SEE PROFILE

Gabriel Torreão

Federal University of Pernambuco

6 PUBLICATIONS **34** CITATIONS

SEE PROFILE

Jian Fan

HP Inc.

69 PUBLICATIONS **2,408** CITATIONS

SEE PROFILE

A New Method for Text-Line Segmentation for Warped Documents

Daniel M. Oliveira[1], Rafael D. Lins[1], Gabriel Torreão[1], Jian Fan[2], and Marcelo Thielo[3]

[1] Universidade Federal de Pernambuco, Departamento de Eletrônica e Sistemas.
Av. Prof. Luiz Freire, s/n, Cidade Universitária
50740-540, Recife, PE, Brasil
{rdl,daniel.moliveira,gabriel.dsilva}@ufpe.br
[2] Hewlett-Packard Labs, Palo Alto, USA
jian.fan@hp.com
[3] Hewlett-Packard Labs, Porto Alegre, Brazil
marcelo.resende.thielo@hp.com

Abstract. Bound documents either scanned or captured with digital cameras often present a geometrical warp that makes text-lines curled. The identification of text-lines is one of the steps for document de-warping when only a single image is available. This paper presents a new method for text-line segmentation. It is based on a simple, but effective, skew detector proposed by Ávila-Lins and simplifies the idea of coupled snakes introduced by Bukhari to a moving parallel line regression. The proposed method performed better than the best of the similar algorithms in the literature.

Keywords: Text-line segmentation, document de-warping, layout analysis.

1 Introduction

The digitalization of bound documents, such as books, either performed by flatbed scanners or digital cameras often yields images that exhibit a geometrical distortion in the region close to the book spine. Such distortion not only makes more difficult the document reading for humans, but also degrades OCR performance. Text-line envelope segmentation is one of the pre-processing steps for many algorithms. The segmentation process can be accompanied of a baseline and/or mean line estimation, Figure 1 illustrates these typographic lines and others.

Bukhari, Shafait, and Breuel [9] introduce the concept of baby snakes for extraction of text-lines. Later on, they use coupled *snakelets* [7] for the same purpose. Finally, in references [8][10] they obtain text-lines by ridges detection on grayscale images. Coupled snakes are used for base/mean line estimation. De-warping is done by calculating y-coordinates using upper/lower neighboring lines followed by a perspective correction.

Stamatopoulos and his colleagues [6] detect text-lines and estimate a document 3D model by approximating the border lines with a polynomial of degree three. Fu *et al*

A. Campilho and M. Kamel (Eds.): ICIAR 2010, Part II, LNCS 6112, pp. 398–408, 2010.

Fig. 1. Font typographic lines following reference [14]

[2] estimate border points fitting them in 3D-cylinder model. Masalovitch and Mestetskiy [1] estimate the spaces between lines; a bezier patch is built and followed by de-warping procedure.

This paper proposes new text-line segmentation method. It borrows ideas from a simple but effective skew detector proposed by Ávila-Lins [12] and coupled snakes introduced by Bukhari, Shafait, and Breuel [7] and can be used regardless document orientation. Reference [4] uses the new method presented herein for de-warping scanned documents. It is organized as follows. Section 2 presents details of the new algorithm. Section 3 shows that the proposed algorithm benchmarked on the CBDAR 2007 de-warping contest test-set achieved an accuracy rate of 91.10% with an under segmentation rate of 1.81%, while the performance of algorithm by Bukhari, Shafait, and Breuel [7], the best algorithm in the literature yields 89.65% and 3.30%, respectively. Section 4 concludes this works.

2 Segmenting Text-Lines

The proposed algorithm improves the Ávila-Lins skew detection scheme [3] for text-line segmentation which is summarized below as illustrated in Figure 2. Black and white images are assumed as input for the algorithm.

1. Component labeling transforming components as enclosing blocks (Figure 2.a);
2. For each unvisited block B do:
   a. Locate the nearest unvisited neighbor block N of block B (Figure 2.b);
   b. Group a text-line starting from blocks B and N (Figure 2.c-d) forming upper/lower or right/left lines;
   c. Detect the skew angle and landscape/portrait orientation of the document;
   d. Detect the up-down orientation of the document;
3. Detect total document rotation;

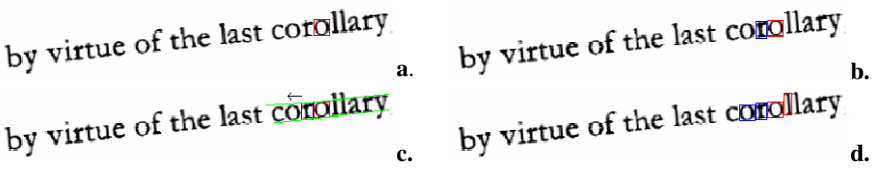

Fig. 2. Ávila-Lins skew detection [3]

This (in C language) algorithm is fast. For a 200dpi scanned image, it takes 115ms on a Pentium IV of 2.4GHz and 512MB of RAM with accuracy of 98%. Despite the good results in skew detection, text-line segmentation requires a more robust approach.

2.1 Letter/Line Properties

Let *V* be the vector formed by the extreme points of a line. The slope of *V*, called the *V* angle, is considered to be the angle that the text-line forms with the horizontal line. If the absolute value for the x-component is greater than the y-component of this vector, the line is considered to be horizontal, otherwise it is vertical. The Text-line length is set to *V* length.

Table 1. Letter properties definition

Orientation / Letter property	Horizontal	Vertical
Letter case height	Block height	Block width
Letter case width	Block width	Block height
Letter case top point	Upper middle point	Left middle point
Letter case bottom point	Lower middle point	Right middle point

One may notice that letter properties are subject to orientation, thus term "block" is used for the "letter case", the enclosing box relative to image; term "letter" depends on the document orientation as described in Table 1. The discrimination between the character height and width is useful as the width is less stable than the height due to: variable font width values and character merging caused by digitization and/or binarization (e.g. see "precondition" and "width" of Fig. 3).

For the steps presented in the next section, some terms are underlined in a high level language with a more precise definition below; the ratio function is defined by (1):

- Small block – Box where width and height have less than 6 pixels;
- Similar size letters – Letters N and M have similar size if the $ratio(N_{height}, M_{height}) \geq 0.6$ and $ratio(N_{width}, M_{width}) \geq 0.1$.
- Parallel lines with offset – an offset of 40% relative to y-axis intercept (i.e. $|b_t - b_d|$) is added to top/down lines;
- Smaller than window mean widths/heights – Letter N and properties window W with $ratio(N_{height}, W_{\overline{height}}) < 0.6$ and $N_{width} \leq W_{\overline{width}}$.
- Maximum distance between letters – 2.50 times window mean height;
- Search for text-line upwards/downwards – search range are limited to 3 times the height of a letter;

$$ratio(a, b) = min(a, b) / max(a, b) \ . \tag{1}$$

2.2 The New Approach

The main idea of the new method proposed here is to group together characters with same properties by "walking" through the document to form a text-line. Instead of coupled snakes [7], moving parallel straight lines are used; Section 2.2.1 explains how to obtain them. As the warp level may distort character sizes, a moving letter window is used while a text-line is formed; A window of length of 7 components was used herein. Listing 1 summarizes the whole procedure.

Listing 1

1. Label components transforming them as enclosing boxes;
2. Remove small blocks or if a block encloses another totally;
3. For each block B do (term "block" is used; orientation is not available):
   a. Locate the nearest neighbor block N of block B;
   b. If N and B have similar sizes, place them in $Q_{NEIGHBORS}$ priority queue, with $priority = ratio(N_{width}, B_{width}) + ratio(N_{height}, B_{height})$;
4. While $Q_{NEIGHBORS}$ is not empty
   a. Pull-out neighbors (B and N) from $Q_{NEIGHBORS}$
   b. If any of B or N was visited go to step 4
   c. Create new text-line TL and add (B and N) neighbors
   d. Search letters between in B to N direction (width/height are orientation dependent):
      i. Create a moving properties window;
      ii. Search for a letter using parallel lines;
      iii. If a letter is found add it to TL if:
         1. It has similar size when compared to moving mean of widths/heights or if it is smaller than window mean widths/heights;
         2. The box center is between parallel lines with offset;
         3. The distance between the last letter and new one is less than the maximum distance between letters;
      iv. Add the letter onto neighbor candidate list on TL if conditions 1-2 above are met and the third is not;
      v. If the letter was added to TL, append it to properties window if it is not smaller than window mean widths/heights; this prevents from adding small components (e.g. accents, punctuation marks) to parallel regression;
   e. Execute previous step for direction N to B;
   f. Place text-line in $Q_{TEXTLINE}$ priority queue, with $priority = textline\ length$
   g. Mark all letters on the new text-line as visited;
5. Remove text-lines whereas its angle is 90° apart most common text-line angle;
6. While $Q_{TEXTLINE}$ is not empty (process bigger text-lines first)
   a. Pull-out text-line TL from $Q_{TEXTLINE}$
   b. If TL was merged go to step 6
   c. For each letter on TL search for text-line upwards and add it to UTL;
   d. For each letter on TL search for text-line downwards and add it to DTL;

e. If there are any letter in common between UTL and DTL then
   i. Mark current TL text-line as an invalid text-line;
   ii. Delete it from text-line list;
   iii. Go back to step 6;
f. Merge two text-lines in UTL if one have a candidate neighbor of the other and vice-versa;
g. Add new textlines to $Q_{TEXTLINE}$ with $priority = textline\ length$
h. Repeat steps 6.f-g for DTL;

7. Remove text-lines where letter count is less the moving window;
8. Remove text-lines if it contains a letter on 10% of image border (mark them as noise);
9. For each text-line
   a. Calculate a simple moving linear regression [12] for top/down points
   b. Compute corresponding point in the line and its deviation error
10. Set top or down points as baseline whether the set with less error;

Figure 3 shows an example of the execution of step 4, where letters with boxes belong to the moving window. Figure 4 shows on upper left finding parallel lines and neighbor candidates between lines on upper right, on bottom shows merging result. Figure 5 shows the execution of step 7-8, with top and down sample points in gray and baseline points in black. The Method described herein can also be used to estimate document orientation using text-line angle histogram.

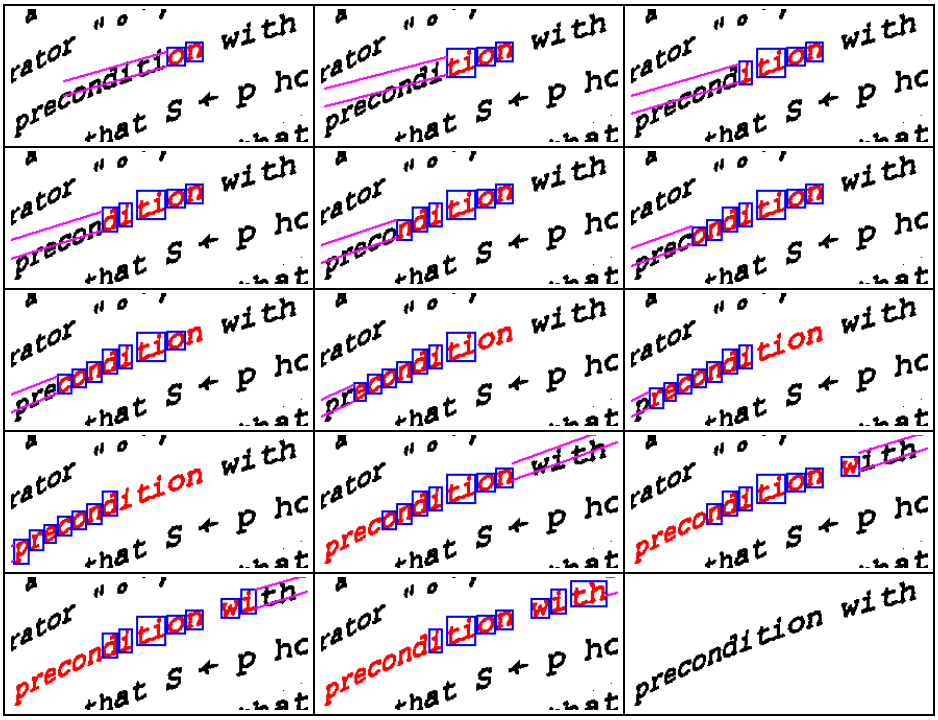

Fig. 3. Example of text-line formation by the new method

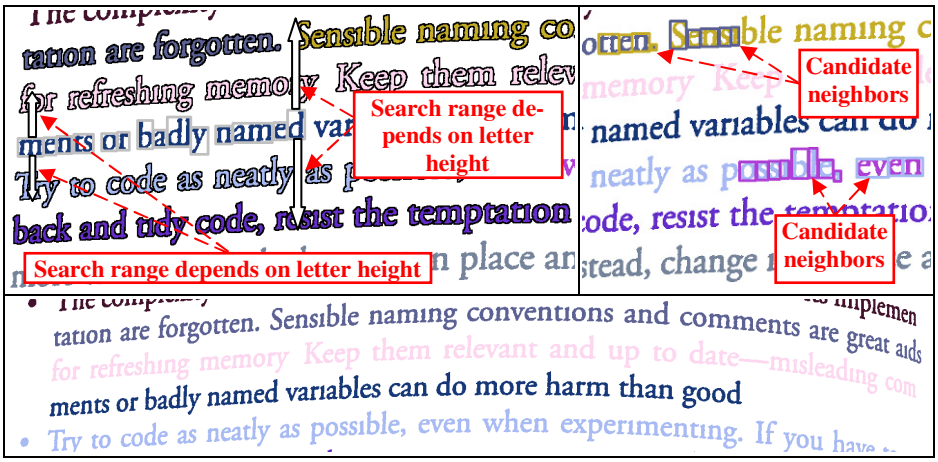

Fig. 4. Text-lines merging procedure: (upper left) parallel lines; (upper right) candidate neighbors; (bottom) merging result

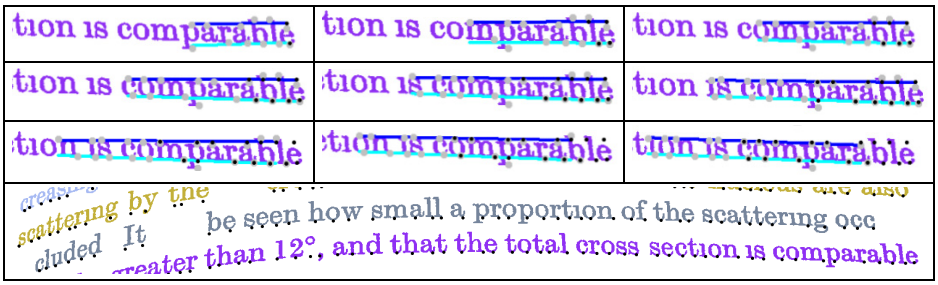

Fig. 5. Baseline estimation with simple moving simple linear regression

2.1.1 Parallel Line Regression

The aim of parallel lines regression is to minimize the error function of equation (2). Where m is the slope which is the same for both lines; b_t and b_d parallel lines intercept for top and bottom lines, respectively; (x_i, y_i) and (x_k, y_k) are the top and bottom samples points, respectively; N is the number of pairs of sample points.

$$E(m, b_t, b_d) = \sum_{i=1}^{N} (m \times x_i + b_t - y_i)^2 + \sum_{k=1}^{N} (m \times x_k + b_d - y_k)^2 \quad (2)$$

Making $\partial E/\partial m = \partial E/\partial b_t = \partial E/\partial b_d = 0$, results in eqs. (3)-(5).

$$m\left(\sum_{i=1}^{N} {x_i}^2 + \sum_{k=1}^{N} {x_k}^2\right) + b_t \sum_{i=1}^{N} x_i + b_d \sum_{k=1}^{N} x_k = \sum_{i=1}^{N} y_i x_i + \sum_{k=1}^{N} y_k x_k \quad (3)$$

$$m\sum_{i=1}^{N} x_i + b_t N = \sum_{i=1}^{N} y_i \tag{4}$$

$$m\sum_{k=1}^{N} x_k + b_d N = \sum_{k=1}^{N} y_k \tag{5}$$

Using Cramer's rule, m, b_u and b_l values are obtained in eqs. (6)-(9).

$$\Delta = \begin{vmatrix} \left(\sum_{i=1}^{N} {x_i}^2 + \sum_{k=1}^{N} {x_k}^2\right) & \sum_{i=1}^{N} x_i & \sum_{k=1}^{N} x_k \\ \sum_{i=1}^{N} x_i & N & 0 \\ \sum_{k=1}^{N} x_k & 0 & N \end{vmatrix} \tag{6}$$

$$m = \frac{\Delta m}{\Delta};\ \Delta m = \begin{vmatrix} \left(\sum_{i=1}^{N} y_i x_i + \sum_{k=1}^{N} y_k x_k\right) & \sum_{i=1}^{N} x_i & \sum_{k=1}^{N} x_k \\ \sum_{i=1}^{N} y_i & N & 0 \\ \sum_{k=1}^{N} y_k & 0 & N \end{vmatrix} \tag{7}$$

$$b_t = \frac{\Delta b_t}{\Delta};\ \Delta b_t = \begin{vmatrix} \left(\sum_{i=1}^{N} {x_i}^2 + \sum_{k=1}^{N} {x_k}^2\right) & \left(\sum_{i=1}^{N} y_i x_i + \sum_{k=1}^{N} y_k x_k\right) & \sum_{k=1}^{N} x_k \\ \sum_{i=1}^{N} x_i & \sum_{i=1}^{N} y_i & 0 \\ \sum_{k=1}^{N} x_k & \sum_{k=1}^{N} y_k & N \end{vmatrix} \tag{8}$$

$$b_d = \frac{\Delta b_d}{\Delta};\ \Delta b_d = \begin{vmatrix} \left(\sum_{i=1}^{N} {x_i}^2 + \sum_{k=1}^{N} {x_k}^2\right) & \sum_{i=1}^{N} x_i & \left(\sum_{i=1}^{N} y_i x_i + \sum_{k=1}^{N} y_k x_k\right) \\ \sum_{i=1}^{N} x_i & N & \sum_{i=1}^{N} y_i \\ \sum_{k=1}^{N} x_k & 0 & \sum_{k=1}^{N} y_k \end{vmatrix} \tag{9}$$

3 Results

Reference [11] compares the methods presented in [7][9][10] using CBDAR de-warping dataset [5]. Herein, the same comparison methodology, described in [13], is used. The ground truth and hypothesized (processed) image have each line painted using a different color. Their similarity is compared by a pixel-correspondence graph, where each node represents a text line in both images; the edges are text-line pixels that are shared between them where the weight is the total number of pixels shared. Black and white are not text-line colors; they stand for non-textual (noise) and background pixels, respectively. An incoming edge is significant if $w_i/P \geq T_r$ and $w_i \geq T_a$, where w_i is the weight of the edge; P is the total number of node pixels; T_r and T_a are the relative and absolute thresholds. The following parameters are computed (copied from [11]).

- Number of ground truth lines (N_g) – total number ground truth lines in the whole database.
- Total correct segmentation (N_{o2o}) – the number of one-to-one matches between the ground-truth components and the segmentation components.
- Total over segmentations (N_{oseg}) – the number of significant edges that ground truth lines have, minus the number of ground truth lines.
- Total undersegmentations (N_{useg}): the number of significant edges that segmented lines has minus the number of segmented lines.
- Oversegmented components (N_{ocomp}): the number of ground truth lines having more than one significant edge.
- Undersegmented components (N_{ucomp}): the number of segmented lines having more than one significant edge.
- Missed components (N_{mcomp}): the number of ground truth components that matched the background in the hypothesized segmentation.
- False alarms (N_{falarm}): the number of components in the hypothesize segmentation that did not match any foreground component in the ground-truth segmentation.
- % correct segmentation (P_{o2o}) – N_{o2o}/N_g
- % oversegmented text-lines (P_{ocomp}) – N_{ocomp}/N_g
- % undersegmented text-lines (P_{ucomp}) – N_{ucomp}/N_g
- % missed text-lines (P_{mcomp}) – N_{mcomp}/N_g

Table 2 shows results of new algorithm and [7][9][10] (copied from [11] until writing of this article), where G-ridges and B-ridges stands for [10] the segmentation in grayscale and binary images, respectively. The proposed method has the best performance for under segmentation, false positives and correct segmentation figures. No parallel line merging was registered, lines where merged if they were aligned but belong to other column such as in Figure 6. Despite the highest missed components among other algorithms, missed lines are suppressed by matched ones when it is used together with a de-warping method. An example of successful (P_{o2o}=100%) processing can be seen in Figure 7 with noisy pixels in black. The proposed algorithm proved also to be fast, running in 8.75s with Java implementation over Windows Vista Business on a Dell D531 3GB.

Table 2. Algorithms comparison metrics

Algorithm	N_g	N_s	P_{o2o}	P_{ocomp}	P_{ucomp}	P_{mcomp}	N_{oseg}	N_{useg}	N_{falarm}
New	3091	2924	**91.10%**	21.71%	**1.81%**	4.43%	682	**57**	**785**
B-Snakes	3091	3371	87.58%	5.79%	2.91%	0%	294	117	13199
Ridges (G)	3091	3045	89.10%	3.53%	3.85%	0.91%	115	131	1186
Ridges (B)	3091	3115	89.65%	4.40%	3.30%	0.29%	144	110	2183
C-Snakes	3091	2799	78.26%	**1.26%**	9.06%	0%	**39**	359	3251

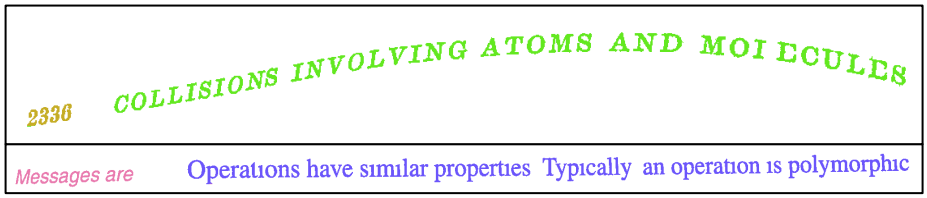

Fig. 6. Line merging examples

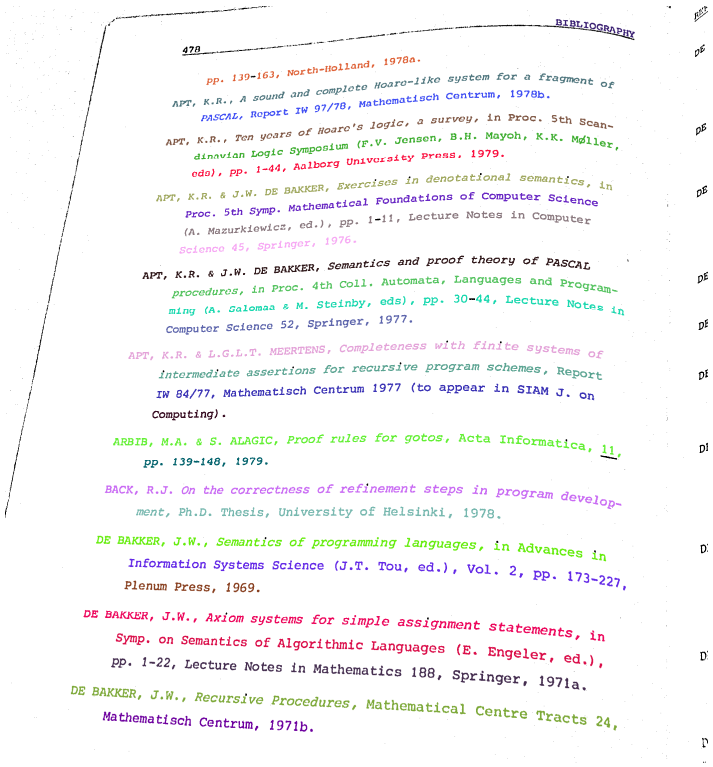

Fig. 7. Example of successful processing

4 Conclusions

A new algorithm for text-line segmentation is presented. It outperforms the other state-of-the-art algorithms with 91.10% of accuracy and 1.81% of under segmentation rates. It can automatically detect text baselines with any orientation, proving also to be fast running in 8.75s with Java implementation for CBDAR 2007 images. The new process was used successfully in correcting the binding distortion in scanned books [4] where the document orientation is arbitrary.

Acknowledgments

The authors like to thank Syed Bukhari, Thomas Breuel and Faisal Shafait for providing page segmentation performance evaluation program source code and for discussions on the subject.

The research reported herein was sponsored by a MCT-Brazilian Government R&D Grant and CNPq funding.

References

[1] Masalovitch, A., Mestetskiy, L.: Usage of continuous skeletal image representation for document images de-warping. In: Proceedings of International Workshop on Camera-Based Document Analysis and Recognition, Curitiba, pp. 45–53 (2007)

[2] Fu, B., Wu, M., Li, R., Li, W., Xu, Z.: A model-based book de-warping method using text line detection. In: 2nd Int. Workshop on Camera-Based Document Analysis and Recognition, Curitiba, Brazil (September 2007)

[3] Ávila, B.T., Lins, R.D.: A fast orientation and skew detection algorithm for monochromatic document images. In: Proceedings of the ACM Symposium on Document Engineering, Bristol, UK, pp. 118–126 (2005)

[4] Lins, R.D., Oliveira, D.M., Torreão, G., Fan, J., Thielo, M.: Correcting Book Binding Distortion in Scanned Documents. In: Campilho, A., Kamel, M. (eds.) ICIAR 2010, Part II. LNCS, vol. 6112, pp. 355–365. Springer, Heidelberg (2010)

[5] Shafait, F., Breuel, T.M.: Document Image De-warping Contest. In: 2nd Int. Workshop on Camera-Based Document Analysis and Recognition, CBDAR 2007, Brazil, September 2007, pp. 181–188 (2007)

[6] Stamatopoulos, N., Gatos, B., Pratikakis, I., Perantonis, S.J.: A two-step de-warping of camera document images. In: Proceedings 8th IAPR Workshop on Document Analysis Systems, Nara, Japan, pp. 209–216 (2008)

[7] Bukhari, S.S., Shafait, F., Breuel, T.M.: Coupled snakelet model for curled textline segmentation of camera-captured document images. In: Proceedings 10th International Conference on Document Analysis and Recognition, Barcelona, Spain, pp. 61–65 (2009)

[8] Bukhari, S.S., Shafait, F., Breuel, T.M.: Ridges based curled textline region detection from grayscale camera-captured document images. In: Jiang, X., Petkov, N. (eds.) Computer Analysis of Images and Patterns. LNCS, vol. 5702, pp. 173–180. Springer, Heidelberg (2009)

[9] Bukhari, S.S., Shafait, F., Breuel, T.M.: Segmentation of curled textlines using active contours. In: Proceedings 8th IAPR Workshop on Document Analysis Systems, Nara, Japan, pp. 270–277 (2008)

[10] Bukhari, S.S., Shafait, F., Breuel, T.M.: Textline information extraction from grayscale camera-captured document images. In: Proc. The 13th International Conference on Image Processing, Cairo, Egypt (2009)
[11] Bukhari, S.S.: Technical Report: Performance Evaluation and Benchmarking of Three Curled Textline Segmentation Algorithms. IUPR Techinal Report, Kaiserslautern (2010)
[12] Wolfram Resarch. Least Squares Fitting, http://mathworld.wolfram.com/LeastSquaresFitting.html (accessed January 15, 2010)
[13] Shafait, F., Keysers, D., Breuel, T.M.: Performance evaluation and benchmarking of six page segmentation algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence 30(6), 941–954 (2008)
[14] Naylor, M.: Typographic line terms, http://en.wikipedia.org/wiki/File:Typography_Line_Terms.svg