

A Statistical approach to line segmentation in handwritten documents

Manivannan Arivazhagan, Harish Srinivasan and Sargur Srihari
Center of Excellence for Document Analysis and Recognition (CEDAR)
University at Buffalo, State University of New York
Amherst, New York 14228
{m44a,hs32,srihari}@cedar.buffalo.edu

ABSTRACT

A new technique to segment a handwritten document into distinct lines of text is presented. Line segmentation is the first and the most critical pre-processing step for a document recognition/analysis task. The proposed algorithm starts, by obtaining an initial set of candidate lines from the piece-wise projection profile of the document. The lines traverse around any obstructing handwritten connected component by associating it to the line above or below. A decision of associating such a component is made by (i) modeling the lines as bivariate Gaussian densities and evaluating the probability of the component under each Gaussian or (ii) the probability obtained from a distance metric. The proposed method is robust to handle skewed documents and those with lines running into each other. Experimental results show that on 720 documents (which includes English, Arabic and children's handwriting) containing a total of 11,581 lines, 97.31% of the lines were segmented correctly. On an experiment over 200 handwritten images with 78,902 connected components, 98.81% of them were associated to the correct lines.

1. INTRODUCTION

Line segmentation is the first and the most critical pre-processing step for a document recognition/analysis task. Once the lines of handwritten text has been separated, these can then be subjected to word segmentation, word recognition, and other indexing steps necessary for document recognition/retrieval operations. Presence of skew, lines running into each other and absence of any knowledge about the content of the document poses a lot of difficulties for line segmentation. Many approaches such as,¹⁻³ use global/piece-wise projection profile but fail to recognize when the projection profile information is useful and when it is not. The method explained in,⁴ requires parameters to be set, according to the type of handwriting. The Cut Text Minimization method,³ fails to detect short lines and those which do not begin in the beginning of the document. Skew detection methods^{5,6} and baseline estimation methods,^{4,5} are not flexible to capture the variation in handwriting. The method in⁷ assumes that each connected component belongs to one line, which is not the case in documents with lines running into each other. Clustering algorithm based on heuristics have been used for line segmentation in the CEDAR-FOX system.^{8,9} The system's algorithm was originally designed for handwritten postal envelopes but the heuristics do not generalize well to the variations encountered in other handwritten documents.

This paper proposes a novel approach to line segmentation involving the use of bivariate Gaussian densities to model lines. In this way, the skew of the document is captured by the covariance of the Gaussian density. Piece-wise projection profiles are used to obtain an initial set of candidate line starting positions. The line drawn traverses around any obstructing handwritten component by associating it to the above or the below line. The algorithm also automatically recognizes lines running into each other and cuts through at the most appropriate position. Figures 1,2,3 shows three kinds of sample documents on which the line segmentation is performed. The rest of the paper is organized as follows. Section 2 describes each step of the algorithm in detail. Experiments and results are discussed in section 3, which is followed by conclusion in section 4.

2. ALGORITHM

A high level description of the steps of the algorithm is given below.

1. Thresholding and chain code document representation.

From Nov 10, 1999
 Jim Elder
 829 Loop Street, Apt 300
 Allentown, New York 14707
 To Dr. Bob Grant
 602 Queensberry Parkway
 Omar, West Virginia 25638
 We were referred to you by Xena Cohen at the University Medical Center. This is regarding my friend, Kate Zack.
 It all started around six months ago while attending the "Ruby" Jazz Concert. Organizing such an event is no picnic, and as President of the Alwin Association, a co-sponsor of the event, Kate was overworked. But she enjoyed her job and did what was required of her with great zeal and enthusiasm.
 However, the extra hours affected her health; halfway through the show she passed out. We rushed her to the hospital, and several questions, X-rays and blood tests later, we were told it was just exhaustion.
 Kate has been in very bad health since. Could you kindly take a look at the results and give us your opinion?
 Thank you!
 Jim

Figure 1. Line Segmentation on a simple image

2. Get initial set of candidate lines.
3. Line drawing algorithm
 - (a) All lines are drawn parallelly from left to right and modeled using bi-variate Gaussian densities.
 - (b) Any obstructing handwritten component is associated with the line above or below by making a probabilistic decision.
 - (c) The lines are guided by the piece-wise projection profile if available.

Each step is explained in detail below.

2.1. Thresholding and document representation

Grayscale images are thresholded using Otsu's thresholding algorithm.¹⁰ On tough images such as historical manuscripts, the algorithm automatically switches to an adaptive implementation of the thresholding algorithm by looking at the histogram and the entropy of the document. The resulting binary image is traced to obtain the chain code representation of the exterior and the interior contours. The chain-code is a lossless efficient representation of the document wherein one can switch between chain-code and the document image without loss of information. This representation enables fast labeling of every connected component uniquely. The document is now ready for the line segmentation algorithm.

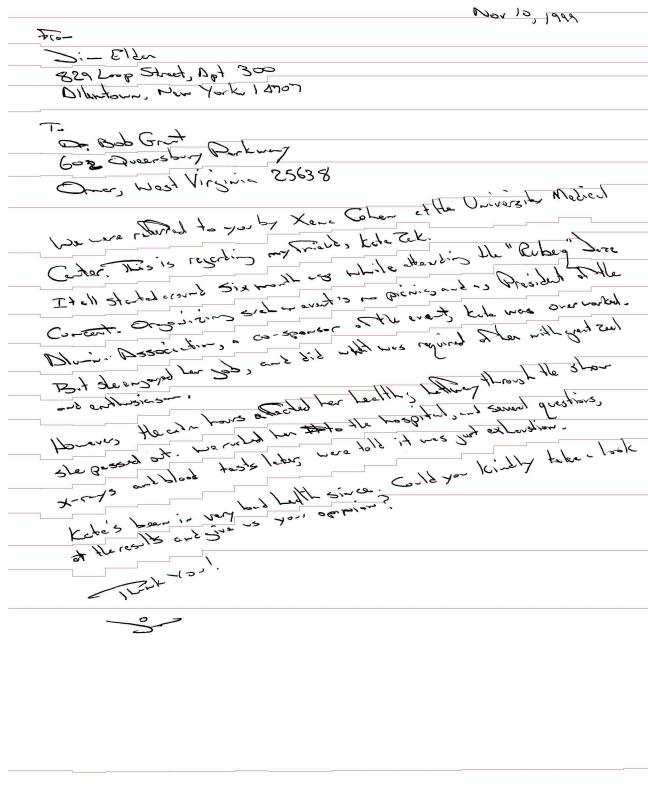


Figure 2. Line Segmentation on a skewed image

2.2. Initial set of candidate lines

The document is divided across its width into chunks of 5% each. Figure 4 shows the projection profile of foreground pixels in each chunk. The projection profile of the first 25% is used to obtain the starting positions of the lines. The starting positions of the candidate lines can be effectively obtained by considering the valleys in the *smoothed* projection profile. Smoothing is done to remove spurious peaks and valleys in the projection profile, by using a simple moving average filter with window of length 5. A valley in a projection profile, is the lowest point between two peaks. This corresponds to the least number of foreground pixels highlighting the point of separation of the two lines. *Smoothed* Projection profile are also obtained for each of the 5% chunks mentioned above. The valleys in *smoothed* projection profiles of the finer 5% chunks help to guide the line further in the right orientation. Let us denote the chunks of *smoothed* projection profiles from left to right as $\psi_1, \psi_2 \dots \psi_N$, where $N = 20$ is the total number of chunks. The *smoothed* projection profile of the first 25% of the document is used to start drawing lines until the end of ψ_1 . For instance, if we find 15 valleys in the *smoothed* projection profile (suggesting the presence of 15 lines) of the first 25% of the document, we start drawing 15 lines at the respective valleys until the end of ψ_1 . That is, the valleys found in ψ_1 is not used. There after, valleys found in $\psi_i, i \in \{2, \dots, 20\}$, is used. The complete set of candidate lines are drawn using the below mentioned rules.

1. Connect a valley from ψ_{i+1} to the closest valley in $\psi_i, i \in \{1 \dots N-1\}$ (Figure 5(a)). If two or more valleys from ψ_{i+1} are connected to the same valley in ψ_i , retain the closest pair and reject the rest. Note that the number of valleys in adjacent chunks are not necessarily the same.
2. There can still remain a few valleys in ψ_i not connected to any in ψ_{i+1} . In such cases, line is continued to be drawn straight from the valley in ψ_i . (Figure 5(b))

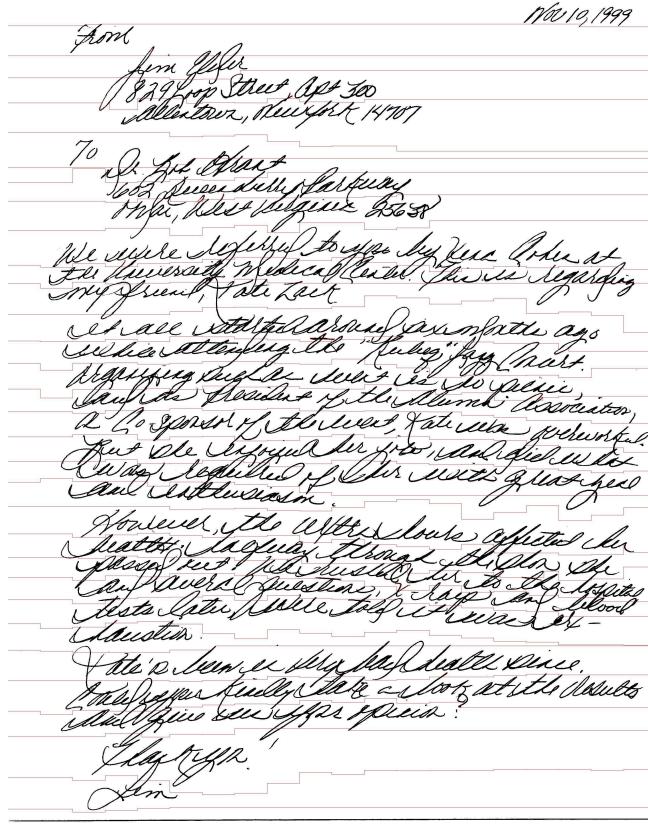


Figure 3. Line Segmentation on a complex image

Figure 5(a) and 5(b) shows the above mentioned rules. Figure 6 show the final result of obtaining the initial set of lines. Lines which have height larger than 1 standard deviation of the mean height are considered separately. These are those that might truly contain two or more lines within it. Such scenarios can arise when the initial candidate line algorithm, fail to recognize the presence of a line of handwritten text. The height of a line is measured as the maximum distance in pixels between the initially drawn candidate lines. All the peaks from the projection profiles for such lines, are taken for consideration. If a peak in $\psi_{\{i+1\}}$, overlaps with any one in the ψ_i , by an overlap factor of *average height of components*, then, these two peaks correspond to the same line of text. If there is no overlap, this indicates the presence of two lines and additional line is drawn below each peak.

2.3. Line drawing algorithm

Using the initial set of candidate lines described in section 2.2, lines are drawn parallelly from left to right. Any line drawn may be obstructed by a handwritten component as shown in figure 7(a). Here a decision is made to associate this component to the line above or below as depicted in figure 7(b). The method of making the decision is either a (i) Gaussian probability decision discussed in 2.3.1 or (ii) Distance metric discussed in 2.3.2.

2.3.1. Gaussian decision

This section describes how the lines are modeled as bi-variate Gaussian densities and the method of associating an obstructing connected component such as the one shown in figure 7(a) with the line above or below. To make a probabilistic Gaussian decision, the individual lines shown in figure 7(b) are first modelled as bi-variate Gaussian densities using the $\{x, y\}$ co-ordinates of the foreground pixels. The sufficient statistics $\vec{\mu}_A$, $\vec{\mu}_B$, Σ_A and Σ_B (the mean vectors and covariance matrices of the line above and below respectively) are computed recursively

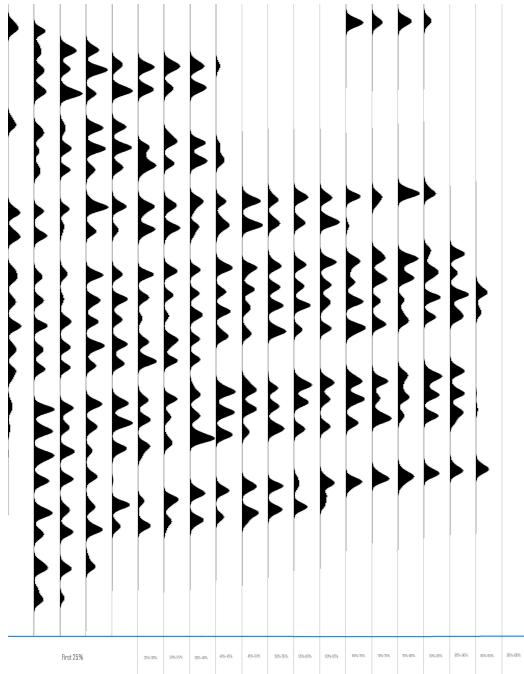


Figure 4. Projection profile of the chunks of the image shown in figure 1. Also is marked with symbols are the first 25% and the individual chunks ψ_i

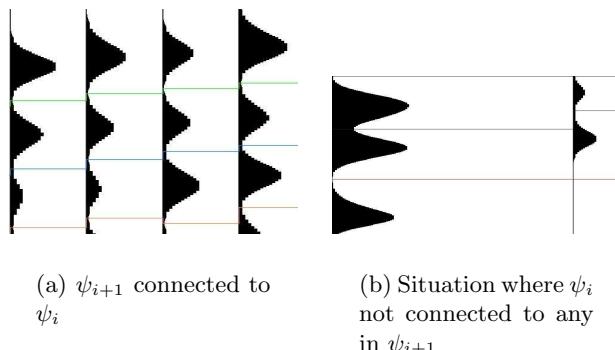


Figure 5. Valley connections (a) All valleys are connected and (b) Few valleys are not connected.

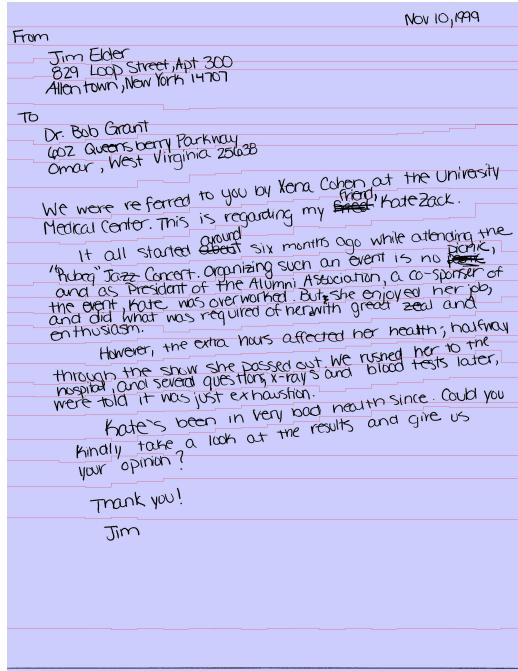
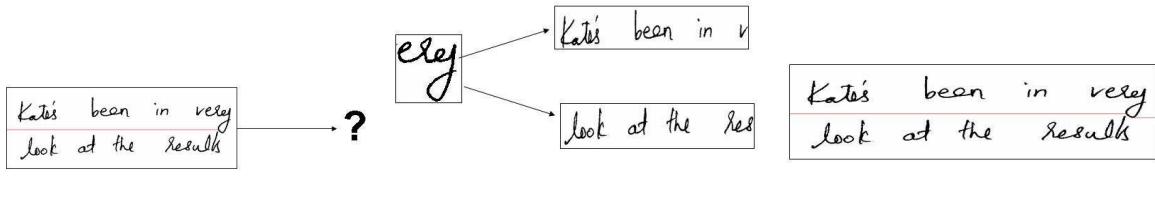


Figure 6. Final set of initial candidate lines



(a) A line drawn is obstructed by the descender of the letter 'y'.

(b) A decision need to be made whether to associate this connected component to the above or the below line.

(c) A line traversing below after deciding that the component belongs to the line above or the below line.

Figure 7. (a) Obstructing component (b) Question of associating the component to the line above or below (c) Line drawn after a decision is made.

by taking one pixel after another. The recursive equations to compute the same for the line above are given in equation 1 and 2. Similarly the parameters are updated for the line below.

$$\vec{\mu}_{A_{(N+1)}} = \frac{N-1}{N} \cdot \vec{\mu}_{A_{(N)}} + \frac{1}{N} \cdot p_{N+1} \quad (1)$$

where N is the total number of pixels already considered, $N+1$ is the index of the new pixel which needs to be included in the update and $p_{N+1} = [x_{N+1}, y_{N+1}]^T$ is the $N+1^{th}$ pixel. The advantage of using the recursive equations is that, as the line grows, the lines can be modeled dynamically much faster by just considering the pixels that were not considered prior.

$$\Sigma_{A_{(N+1)}} = \frac{N-2}{N-1} \cdot \Sigma_{A_{(N+1)}} + \frac{1}{N-1} \cdot (p_{N+1} - \vec{\mu}_{A_{(N+1)}})(p_{N+1} - \vec{\mu}_{A_{(N+1)}})^T \quad (2)$$

The probability of a component to be associated to the line above or below, is computed by evaluating the joint probability of all the pixels of the component, under the Gaussian above and below respectively. In order to compute the joint probability of all the pixels in the component \mathcal{C} comprising of pixels \vec{p} , we make use of the chain rule of probability as shown in equation 3.

$$P(\mathcal{C}|\vec{\mu}_A, \Sigma_A) = P(p_1|\vec{\mu}_A, \Sigma_A) \cdot P(p_2|\vec{\mu}_A, \Sigma_A, p_1) \dots P(p_T|\vec{\mu}_A, \Sigma_A, p_1, p_2, \dots, p_{T-1}) \quad (3)$$

where T is the total number of pixels in the connected component. Each term in the above equation 3 is calculated by evaluating the bi-variate Gaussian density at the pixel p_i . The equation for the bi-variate Gaussian density is given as $P(p_i|\mu, \Sigma) = |2\pi\Sigma|^{\frac{1}{2}}(p_i - \mu)\Sigma^{-1}(p_i - \mu)^T$. After the two probabilities corresponding to the above and below line: $P(\mathcal{C}|\vec{\mu}_A, \Sigma_A)$ and $P(\mathcal{C}|\vec{\mu}_B, \Sigma_B)$ are obtained, the decision to associate the component is made by choosing the line with the larger probability.

2.3.2. Distance metric decision

The distance metric decision is used for the decision making under three conditions:

1. When the obstructing component is present in the first 25% of the document. Here, the number of foreground pixels present in the above and below line are too few to model them as Gaussians with statistical significance.
2. When the strength of evidence of the Gaussian decision is poor. The strength of evidence is measured as the ratio of the probabilities of the above and the below line.
3. No foreground pixels exists in the above/below line to model them as Gaussians.

Let us denote the pixel at which the obstructing component is hit at $p_h = [x_h, y_h]^T$. Now, from this pixel, the contour of the connected component is traversed in two directions. One direction involves moving clockwise(upwards) along the contour, until a pixel with 'y' co-ordinate $y = y_h$ is reached. Similarly, the other direction involves moving anti-clockwise(downwards) with the same rule. Let us define the following:

$$y_u : \text{The pixel of minimum 'y' co-ordinate in the upwards traversal} \quad (4)$$

$$y_d : \text{The pixel of maximum 'y' co-ordinate in the downwards traversal} \quad (5)$$

Compute the probability $p_u = \frac{|y_h - y_u|}{|y_h - y_u| + |y_h - y_d|}$ and $p_d = \frac{|y_h - y_d|}{|y_h - y_u| + |y_h - y_d|}$. The decision to associate the component to the above or below line is done by considering the larger of the two probabilities.

2.4. Line drawing

The valleys in the projection profiles of each chunk ψ_i (mentioned earlier), guide the line drawn to allow as minimal as possible, encounters with obstructing components. The projection profile may not be available or is invalid under two conditions mentioned below.

1. If the valley in ψ_i is not connected to any in ψ_{i+1} . (Figure 5(b))
2. If the distance from the valley at ψ_i to the valley at ψ_{i+1} is larger than the mean height of components.

Table 1 explains how the line is drawn. After a decision is made, the line traverses downwards, if the component was decided to be belonging to the line above, as shown in 7(c) and the line traverses upwards, otherwise.

	Component is hit	Component is <i>not</i> hit
Projection profile information(valleys) present	Traverse the component (upwards or downwards depending upon the decision) until a valley is reached	Line is continued along the initial set of candidate lines
Projection profile information(valleys) <i>not</i> present	Traverse the component (upwards or downwards depending upon the decision) until an extrema(Maxima: y_d (equation 5) or Minima: y_u (equation 4)) is reached	Line is continued straight

Table 1. Line drawing explanation.

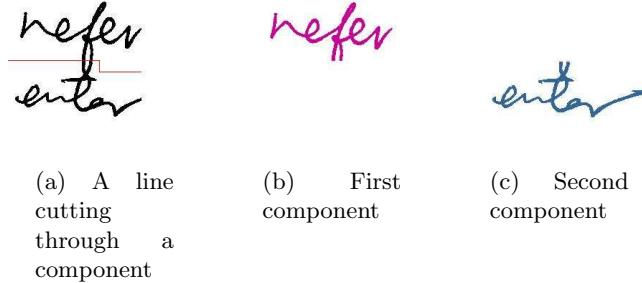


Figure 8. Overlapping Component (a) A line cutting through a component and (b) Component 1 and (c) Component 2.

2.5. Lines running into each other

Documents that contain lines running into each other are characterized by words written in one line overlapping with those in adjacent lines. The connected component in these cases run across two lines and are termed as *overlapping components*. When such components obstruct the line drawn, a decision of cutting through needs to be made. To begin with, such components need to be differentiated from regular obstructing components that do not span across two or more lines. A component is classified as an overlapping component, under any one of the following conditions:

1. If a decision was made to traverse above, and the distance from the minima y_u (equation 4) to the line above is smaller than that to the current line.
2. If a decision was made to traverse below, and the distance from the maxima y_d (equation 5) to the line below is smaller than that to the current line.
3. If the component's minima y_u (equation 4) is lesser than the y-coordinate of the projection profile peak of the previous chunk, and the component's maxima y_d (equation 5) is greater than the y co-ordinate of the projection profile peak of the current chunk.

Figures 8(a),8(b) and 8(c) show an overlapping component being split. The position of cut-through is at the valley in the projection profile. A flood-fill algorithm is run finally to get the final set of line images from the line segmenter's output,which, for instance, is shown in figure 1.

Arabic handwriting is a special case, as it is written from right to left. To make the current algorithm work for an arabic image, the original image is flipped to get its mirror image before sending it to the line segmenter. After the individual line images are obtained by flood-fill, the individual line images are flipped back.

3. EXPERIMENTS AND RESULTS

Table 2 presents a comparison between the previous method^{8,9} and the proposed method. The previous algorithm for line segmentation was a part of the CEDAR-FOX system^{8,9} which used clustering algorithm based on a number of heuristics originally designed for postal envelopes. Table 3 presents a component-level evaluation of

Database	# of lines	Previous Method's Accuracy	Proposed Method's Accuracy
CEDARFOX (300 images)	7201	88.86%	97.01%
ARABIC (120 images)	2030	87.83%	98.62%
Exam Essay (300 images)	2350	86.43%	96.30%

Table 2. Results on 720 handwritten images written by 612 different writers. Note that the average improvement from the previous method is 9.6%

	Total	Number Correct	Accuracy
Number of Lines	4768	4656	97.65%
Number of Components	78902	77958	98.81%
Cut-through	928	690	74.36%

Table 3. Results on 200 handwritten images written by 200 different writers. Note that the proportion of cut through $\frac{928}{78902} = 1.17\%$ is very small.

the results obtained on 200 handwritten images written by 200 different writers.

The line errors were calculated using the following scheme:

1. If even a single component, from one line is associated to another line, it is counted as 2 line errors.
2. If a single line is split into 2 or more lines, then it is counted as 1 line error.
3. If n lines are merged together, then it is counted as n line errors.

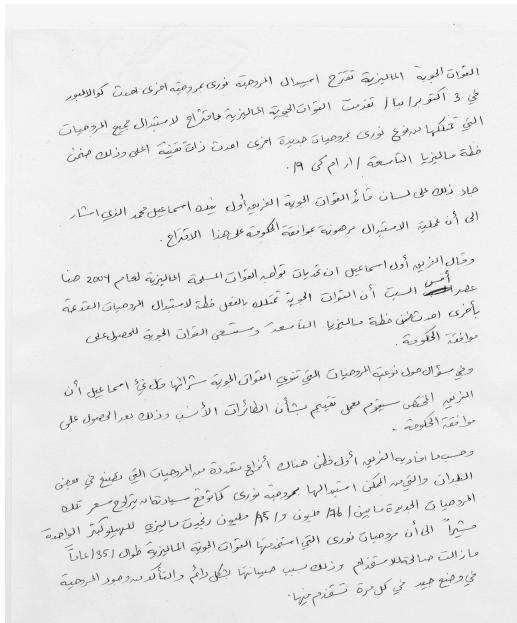
The component errors were calculated considering the following types of errors:

1. Any component from one line associated to another line is counted as 1 component error.
2. If two lines are merged together, the number of the component errors equals the total number of components in both the lines.
3. If one line is split into two lines, the number of component errors equal the number of components in the smaller of the two parts.

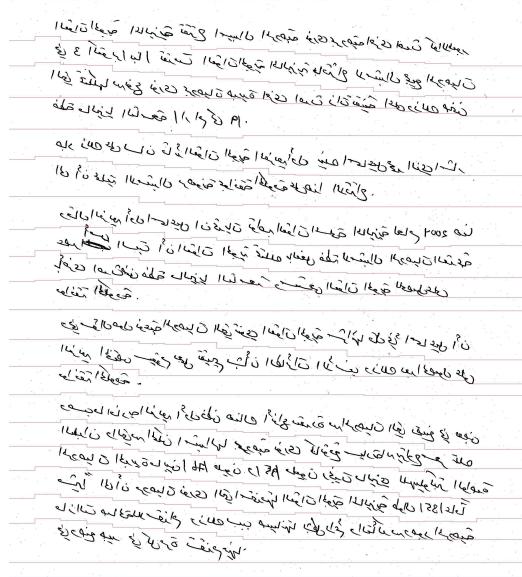
The cut-through accuracy (74.36% mentioned in table 3) corresponds to the proportion of components correctly classified as overlapping components (defined in 2.5). If a normal component is classified as an overlapping component, then it is 1 cut-through error. Cut-through errors are counted only towards component errors and not towards line errors. Since the number of components that are cut-through is very small (1.17%), low accuracy of 74.36% in classification is negligible. However, it has been found that most of these errors are due to the following reasons:

1. A normal component, which spans across two or more lines, is highly prone to be classified incorrectly.
2. A normal component, lying in between two lines of text, is also prone to be classified incorrectly.

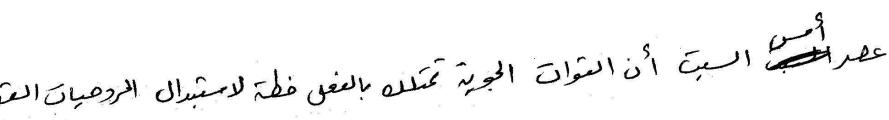
The 300 exam essays mentioned in table 2 were included in the experiments to test the robustness of the algorithm. These essays were written on ruled line paper. They were first automatically preprocessed to remove ruled lines before sending it to the line segmenter. The algorithm was able to segment with an accuracy of 96.3%. Further, the line segmenter proved to be language independent, by achieving a performance of 98.62% on 120 arabic handwritten images. In Arabic, the dots above or below a word are very important in distinguishing one word from another. Most of the dots were found to be associated to the appropriate lines by the proposed algorithm. Figures 9 and 10 shows the line segmentation on a sample arabic image and a sample exam image, respectively.



(a) The original arabic image



(b) Line Segmenter's output; Note the image is the mirror of the true image



(c) A single line image after rotating back; Note the dots being preserved

Figure 9. Line segmentation on an arabic image (a) Original image and (b) Line segmenter's output and (c) single line image.

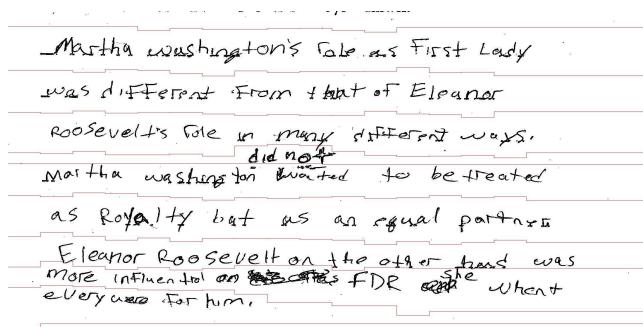


Figure 10. Line Segmentation on a school essay handwritten image

4. CONCLUSION

A novel approach to line segmentation was discussed. The algorithm is robust to handle documents with skew and lines running into each other. The proposed method is based on modeling the lines as bivariate gaussian densities that provide for accurate association of components to the respective lines. The use of piece-wise projection profiles to guide the lines drawn reduces the number of obstructing components. Experimental results show that 98.81% of connected components are associated to the correct lines. The proposed algorithm is superior to existing algorithms that perform the same task. A more intelligent approach to cut an overlapping component is the goal of future work and it suffices to say that a solution to that will yield a nearly perfect line segmentation algorithm.

5. ACKNOWLEDGEMENTS

The authors wish to thank Varun Chakravarthi for his contribution to evaluation of the line segmentation algorithms.

REFERENCES

1. “Segmentation of off-line cursive handwriting using linear programming,” *Pattern Recognition* **31**, pp. 1825–1833, 1998.
2. “An integrated system for handwritten document image processing,” *IJPRAI, International Journal of Pattern Recognition and Artificial Intelligence* **17(4)**, pp. 101–120, 2003.
3. “Handwritten document offline text line segmentation,” *Digital Image Computing: Techniques and Applications, DICTA* , pp. 184–187, December 2005.
4. “Line detection and segmentation in historical church registers,” *Sixth International Conference on Document Analysis and Recognition, Seattle, USA, IEEE Computer Society* , pp. 743–747, September 2001.
5. “A statistically based, highly accurate text line segmentation method,” *Proceedings 5th ICDAR* , pp. 551–554, 1999.
6. “Separating lines of text in free-form handwritten historical documents,” *Second International Conference on Document Image Analysis for Libraries(DIAL)* , pp. 12–23, April 2006.
7. “Text line segmentation in handwritten document using a production system,” *Ninth International Workshop on Frontiers in Handwriting Recognition, IWFHR-9* **26-29**, pp. 245–250, October 2004.
8. S.-H. Cha and S. N. Srihari, “System that identifies writers,” *Proceedings National Conference of American Association of Artificial Intelligence, Austin, TX* , p. pp. 1068, July 2000.
9. S. N. Srihari, B. Zhang, C. Tomai, S. Lee, and Y.-C. Shin, “A system for handwriting matching and recognition,” *Proc. Symp. Document Image Understanding Technology Greenbelt, MD* , pp. pp. 67–75, April 2003.
10. N. Otsu, “A threshold selection method from gray level histograms,” *IEEE Trans. Systems, Man and Cybernetics* **9**, pp. 62–66, Mar. 1979. minimize inter class variance.