

Robust line segmentation for handwritten documents

Kamal Kuzhinjedathu, 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
{krk32,srihari,hs32,}@buffalo.edu

ABSTRACT

Line segmentation is the first and the most critical pre-processing step for a document recognition/analysis task. Complex handwritten documents with lines running into each other impose a great challenge for the line segmentation problem due to the absence of online stroke information. This paper describes a method to disentangle lines running into each other, by splitting and associating the correct character strokes to the appropriate lines. The proposed method can be used along with the existing algorithm¹ that identifies such overlapping lines in documents. A stroke tracing method is used to intelligently segment the overlapping components. The method uses slope and curvature information of the stroke to disambiguate the course of the stroke at cross points. Once the overlapping components are segmented into strokes, a statistical method is used to associate the strokes with appropriate lines.

1. INTRODUCTION

Accurate segmentation of a handwritten document into lines is essential for recognition tasks on the document. Word segmentation and recognition accuracies are severely hurt by inaccurate line segmentation. The problem of line segmentation on a document is non-trivial due to the presence of skewed lines and lines running into each other. Many approaches such as^{2,3,4} 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,⁵ requires parameters to be set, according to the type of handwriting. The Cut Text Minimization method,⁴ fails to detect short lines and those which do not begin in the beginning of the document. Skew detection methods^{6,7} and baseline estimation methods,^{5,6} are not flexible to capture the variation in handwriting. The method in⁸ assumes that each connected component belongs to one line, which is not the case in documents with lines running into each other.

The statistically line segmentation method developed for CEDAR-FOX system¹ is robust and segments documents with skewed lines accurately. Figure 1 shows the result of running the line segmentation algorithm on a complex handwritten document.

However, overlapping lines are handled by choosing the best point to cut the lines apart. Performing such a cut-through results in some parts of components belonging to one line being regarded as belonging to the other. This is undesirable. Ideally, we would like to segment the overlapping component as shown in Figure 2

To address this problem, we present a two-part algorithm. The algorithm tries to approximate the ideal segmentation shown in Figure 2 using stroke tracing techniques. The first part of the algorithm is the statistical line segmentation method presented in.¹ As a result of running this part, we obtain a segmented document and a list of overlapping components that have been cut-through by the algorithm. In the second part of the algorithm, each of these overlapping components in the list is split into its constituent strokes. The strokes are then assigned to either the upper or lower line.

The paper is organized as follows: Section 2 describes the first part of the algorithm. Section 3 discusses in detail how a connected component is decomposed into strokes. Finally, 4 presents the experimental results.

Nov 10, 1999

From
 Jim Foss
 829 Pop Street, Apt 300
 Allentown, PA 18107

To
 Dr. Gus Abbott
 1000 University Blvd
 Allentown, PA 18107

We were referred to you by Gene Cook at
 the University Medical Center. I was regarding
 my friend, Kate Lact.

In all medical records from the past 20
 years regarding the "Katie" case. I
 happened to be seen in the office
 of the President of the Allentown Association
 as the sponsor of the event. Kate was present.
 But she happened to be in the office in her
 car. I was told of her death of heart
 and stroke.

However, the letter book reflects the
 death of Katie through the Allentown
 Association. I was told that she had
 several questions, I had the copies
 sent to her. I had the blood
 donation.

Kate is seen in my past death record.
 I was never fully able to do it. I had
 the blood donation.

Thank you.
 Jim

Figure 1. Line Segmentation on a complex image

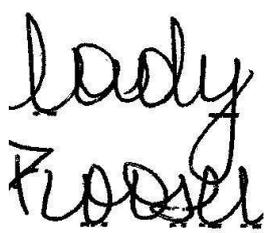
	
Overlapping components	Segmented component

Figure 2. Ideal segmentation of an overlapping component

2. STATISTICAL LINE SEGMENTATION ALGORITHM

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

1. Threshold and create chain code document representation.
2. Get initial set of candidate lines.
3. Line drawing algorithm
 - (a) All lines are drawn in parallel 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.

3. DISENTANGLEMENT OF OVERLAPPING COMPONENTS

The stroke based component segmentation algorithm is invoked once the above algorithm segments the lines and ,in the process, identifies overlapping components.

Once the components which overlap two or more lines have been identified, they have to be broken down into continuous strokes. A continuous stroke is a sequence of pixels in the component which is perceived to have written without lifting the pen off the paper. Once the component has been decomposed into strokes, the probability of each stroke belonging to a line is evaluated and the stroke is associated with the line that assigns it the highest probability.

The following sections describe the steps taken to decompose the component into strokes. The technique is based on the method used for stroke tracing in.⁹

3.1. Derivation of slope and curvature of contour pixels

This step computes the slope and curvature of each pixel in the contour representation of the image. The method requires that the slope and curvature of a pixel capture global information about the contour rather than the local slope and curvature which are susceptible to noise. So the contour is fit with a polynomial curve and the slope and curvature of the pixels are derived from the equation of the curve. The equation of the curve is of the following form:

$$x(t) = a_x.t^2 + b_x.t + c_x \text{ and } y(t) = a_y.t^2 + b_y.t + c_y$$

Due to the complex nature of the contours, it is not possible to accurately model them using one second degree curve. The contour is therefore broken into pieces before fitting each piece with a curve. However, it is crucial that the contour be broken at the right places so as to capture the slope information accurately. The points where the cuts are made are called knots. Identifying proper knots is a non-trivial problem and crucial to correctness of subsequent steps. Here, we choose knots by choosing from a set of candidate points, the points that have desirable knot properties. The maxima and minima of the parametric curves are treated as candidate knots. They are then ordered according the following properties before choosing the best of them.

- Its position in the sequence: Since it is not preferable to cut the sequence into a large number of pieces, Knots which break the sequence almost evenly are preferred.
- Its "sharpness": Sharp turns in the contour correspond to good Knots. The sharpness of a point p is quantified using the following formula.

$$S(p) = |S_1(p) - S_2(p)| \tag{1}$$

where $S_1(p) = p_n - p_{n-1} + \gamma.(p_{n-1} - p_{n-2}) + \gamma^2.(p_{n-2} - p_{n-3})...$ and so on upto the next extrema and $S_2(p) = p_{n+1} - p_n + \gamma.(p_{n+2} - p_{n+1}) + \gamma^2.(p_{n+3} - p_{n+2})...$ and so on upto the next extrema.

The γ factor is so that points closer to the extremum have more influence over the sharpness. $\gamma = 0.5$ is found to work well.

- The height/depth of the maxima/minima.

Piecewise regression is then performed by recursively breaking sequences at the best Knots and fitting the pieces with second degree curves until the desired accuracy is achieved. Figure 3 shows the result of fitting pieces of the contour with second degree curves. Note that the cuts in the curves correctly correspond to sharp turns in the contour.

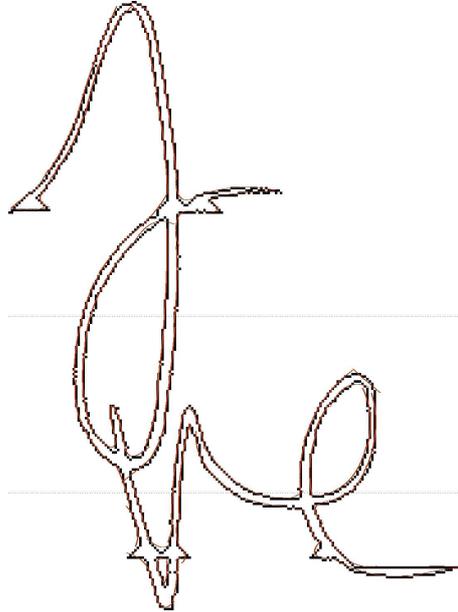


Figure 3. The smooth lines correspond to the curves that are fitting on the contour

Once the equations of the curves that best fit the contours are obtained, the slope and curvature of the pixels are found.

3.2. Formation of cliques and meshes

In,⁹ a clique X_k is defined to be a set of 6 parallel pixels such that $X_k = \{\alpha_{-1k}, \alpha_{-0k}, \alpha_{1k}, \beta_{-1k}, \beta_{-0k}, \beta_{1k}\}$ where $\alpha_x, \beta_x \in \{(x_{ij}, y_{ij})\}$ and α_x is parallel to β_x . At each point, a clique is created if it satisfies the following conditions:

- All α_x is parallel to β_x (tangents and curvature are opposite)
- The stroke thickness for all three points $\{\alpha_x, \beta_x\}$ are comparable
- None of the points are near an end point

A sequence of cliques is called a mesh. Meshes are formed by merging cliques that are close to each other. Figure 4 shows the different meshes formed from the contour pixels.

3.3. Splicing of meshes to form strokes

Meshes correspond to continuous segments of the strokes in the component. The task now is to connect the ends of the meshes formed in the previous step. Meshes are connected to other meshes by predicting their courses in both directions. To achieve this, the centers of the cliques that make up a mesh are modeled using quadratic curves. The technique here is identical to the piecewise regression performed on the contour pixels.

When the center of a mesh is extrapolated, it could 'hit' other meshes or their extrapolations. These hits suggest that the two hitting meshes might have to spliced. When a mesh end hits just one other mesh end,

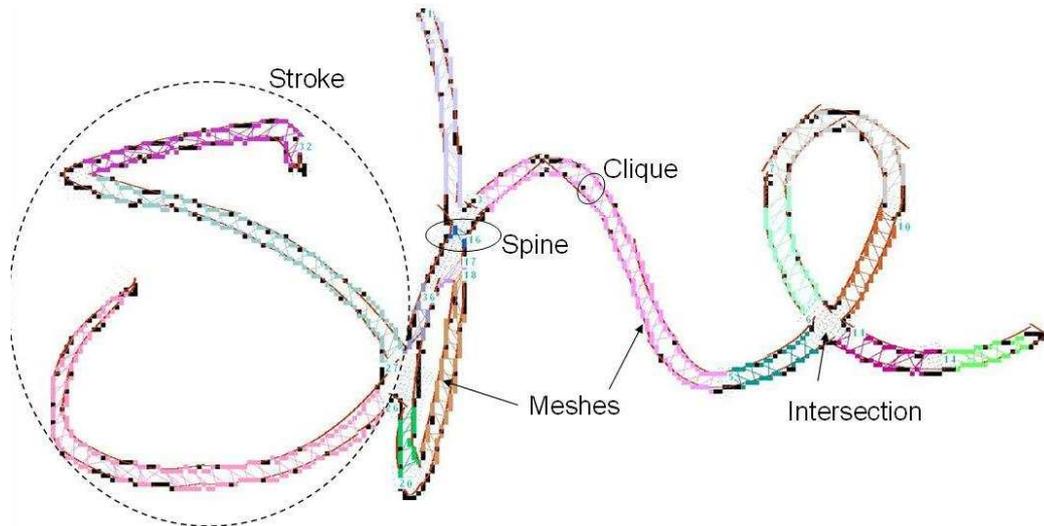


Figure 4. The image shows meshes formed on a connected component. Many meshes can be seen but the component is made of only 2 strokes

the problem is trivial and we can just join the 2 meshes to make a bigger mesh. But it so happens that at intersections, the prediction of the course of a mesh end can 'hit' more than one mesh. The problem of splicing meshes is therefore the problem of connecting a mesh end to the correct mesh end.

More formally, when meshes $m_1, m_2 \dots m_n$ participate in an intersection, we are required to find pairings (m_i, m_j) such that m_i and m_j lie adjacent to each other on a single stroke.

But the problem is further complicated by the fact that when the component has overlapping strokes, one mesh can be formed in the part of the component that two strokes share. Such meshes are called spines. Figure 4 shows such a spine.

3.3.1. Reduction to a classification problem

To solve the above problem, a mesh end has to be connected to the mesh end that it is most 'compatible' with. So when a mesh end e hits mesh ends $e_1, e_2 \dots e_k$, " e " has to be connected to one of these candidates. So we need a mechanism to compare the compatibility of 2 pairs of mesh ends. The classification problem therefore is the following.

Given 2 mesh end pairs $\{(e_{i1}, e_{j1}), (e_{i2}, e_{j2})\}$, label it as:

1. class 1 if (e_{i1}, e_{j1}) is a better compatible pair than (e_{i2}, e_{j2})
2. class 2 if (e_{i2}, e_{j2}) is a better compatible pair than (e_{i1}, e_{j1})
3. class 3 if (e_{i2}, e_{j2}) and (e_{i1}, e_{j1}) are two equally compatible pairs.

By using such a classifier, given a mesh end E , we can choose the best or the best 2 mesh ends to which e has to be connected. Note that class 3 is needed to handle spines.

3.3.2. Artificial neural network classifier

The above classification is done using a 3 layered Artificial Neural Network. The network was trained using truth data created by manually choosing mesh pairs to be spliced. The ANN had 18 hidden neurons. The following features were used for each pair e_i, e_j .

1. The inner product of curvature vectors at the point of intersection of the two meshes

2. The dot product of slope vectors at the point of intersection
3. The ratio of widths of mesh ends
4. The percentage of pixels between the two mesh ends that are black
5. The number of other candidates for e_1
6. The number of other candidates for e_2

3.3.3. Splicing Algorithm

The splicing algorithm takes a set of meshes as input and outputs strokes (spliced meshes). It uses an Artificial Neural Network classifier to achieve this.

The algorithm is as follows:

1. Compute extrapolations of all mesh ends and the resulting 'hits'
2. Splice mesh ends e_i, e_j if e_j is most compatible with e_i only and vice versa. The ANN classifier is used here
3. For each spine
 - (a) Duplicate the spine
 - (b) Connect one copy to one of the 2 mesh ends that it is most compatible with
 - (c) Connect the other copy to the other
4. If any two ends were spliced in this iteration, go back to step 1

The resulting set of long meshes are the strokes that are returned by the algorithm.

3.4. Mapping of strokes to lines

Once the strokes have been identified, the probabilities of the strokes under the bivariate Gaussians (that model the lines above and below) are evaluated and the strokes are assigned to the lines that assign them the greatest probability.

4. RESULTS

Number of images processed	293
Total number of cut-through components	998
Number correctly segmented	727
Success rate	72.8

When the stroke width is too large compared to the length and the curvature information is insufficient to predict the course of the stroke, the segmentation quality is relatively low. The typical cases which are handled well are the ones where the descender from one line overlaps with characters that belong to the one below. Figures 5 and 6 show such cases:

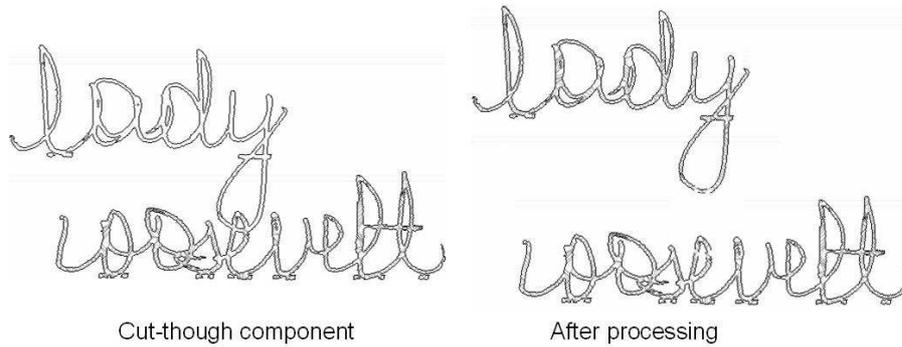


Figure 5. Before and after processing

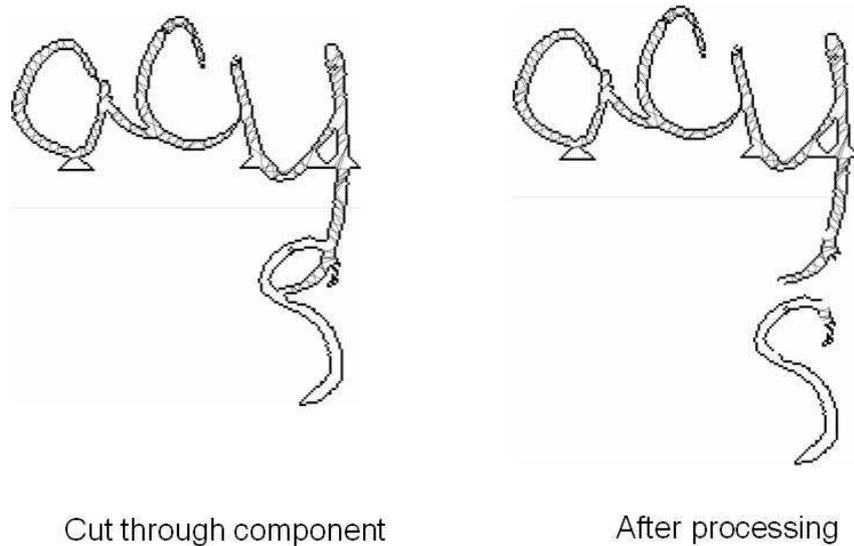


Figure 6. Before and after processing

REFERENCES

1. H. S. Manivannan Arivazhagan and S. Srihari, "A statistical approach to line segmentation in handwritten," 01.
2. "Segmentation of off-line cursive handwriting using linear programming," *Pattern Recognition* **31**, pp. 1825–1833, 1998.
3. "An integrated system for handwritten document image processing," *IJPRAI, International Journal of Pattern Recognition and Artificial Intelligence* **17(4)**, pp. 101–120, 2003.
4. "Handwritten document offline text line segmentation," *Digital Image Computing: Techniques and Applications, DICTA* , pp. 184–187, December 2005.
5. "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.
6. "A statistically based, highly accurate text line segmentation method," *Proceedings 5th ICDAR* , pp. 551–554, 1999.
7. "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.

8. "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.
9. K. B. Gilles F. Houle and M. Shridhar, "Handwriting stroke extraction using a new xytc transform," 01.