Text Line Segmentation for Handwritten Documents Using Constrained Seam Carving

Xi Zhang
School of Computing
National University of Singapore
13 Computing Drive, 113417, Singapore
xizhang@comp.nus.edu.sg

Chew Lim Tan
School of Computing
National University of Singapore
13 Computing Drive, 113417, Singapore
tancl@comp.nus.edu.sg

Abstract—This paper proposes a language-independent method for segmenting text lines from handwritten document images. Our method is based on the seam carving, which has been already used for text line segmentation, but in order to tolerate multi-skewed text lines even in the same document image, we propose a constrained seam carving method, which can constrain energy to be passed along the connected components in the same text line as much as possible. Moreover, our proposed method tries to extract all the text lines by computing the energy map only once. In the experiments, our method is tested on the Greek, English and Indian document images, and get 98.41% FM score.

I. INTRODUCTION

Text line segmentation is a very crucial step for Optical Character Recognition (OCR) [1] and keyword spotting [2] [3], which are used to provide reliable information retrieval through out a large amount of document images. However, unlike the printed documents, which have constrained layout, finite types of fonts, pre-defined sizes of character and uni-skewed text lines, handwritten documents always contain unconstrained writing styles, such as long ascenders or descenders connecting different text lines together, multi-skewed text lines even in the same document image, and small floating strokes, all of which can lead to difficult text line segmentation.

There are two main broad categories of segmenting text lines, one is top-down approaches, and the other is bottom-up approaches. As the name suggests, top-down methods try to estimate the locations of the candidate text lines first. Then the estimation is refined by assigning components to the text lines which they belong to with higher probability, and splitting large components, which touch multiple text lines. On the other hand, bottom-up methods try to find local components first, which are always the connected components (CC), and then group the components together into separate text lines based on different types of grouping algorithms.

For top-down methods, in [4], document images are first divided into separate column chunks, the width of which is 5% of the width of the document. Horizontal projection profiles of foreground pixels are generated for each chunk. Based on the smoothed projection profiles, the valleys in every chunk, where the number of foreground pixels are minimum between two consecutive peaks, are located and used to indicate the positions where two text lines should be separated. The initial estimated text lines are extracted by connecting valleys in each profile with the closest ones in the previous profile. Separating lines are drawn horizontally from left to right, and for unused valleys, separating lines are drawn horizontally at the same position as in the previous profile. When the separating line encounters a component, bi-varient Gaussian densities are used to capture spatial features, and a decision is made to assign the component to the optimal text line, above or below.

Besides, [5] applied a steerable directional filter to get an Adaptive Local Connectivity map (ALCM) of the original document. Using multiple directions of the filters, the convolution results can reflect how likely one text line appears at each position. The estimation is made using the maximum response of the convolutions. In ALCM, large values always correspond to the pixels lying in the dense text regions. So after applying a local adaptive binarization method, regions with dense text lines are left, presenting the entire text lines or partial ones. At last, components crossing multiple text lines are separated and other unassigned components are allocated to the spatially closest text lines.

For bottom-up methods, a Hough Transform based method was proposed in [6]. Document images are binarized and enhanced first, and the connected components (CCs) are extracted. Based on the average height and width of all CCs, CCs are grouped into three exclusive subsets: large components, small CCs, such as accents, and the remaining normal sized CCs, which construct the main body parts of the text lines. For each CC in the third subset, it is partitioned into equally-sized blocks. The Hough transform is applied to the gravity center points of all blocks, and assign a CC to one text line if half of the points are assigned to this text line, according to the accumulator array. In the post processing step, the CCs in the second subset are assigned to the closest text lines, and the CCs in the first subset are either assigned to the text line they only lie on, or separated into different parts, and assigned to individual parts separately.

In [7], the distances of CCs are measured based on a special designed metric using supervised learning, which can enlarge the distance between two neighboring CCs in different text lines, and narrow the distance if they are lying in the same text line. After removing small or large CCs, documents are represented by a graph, each node of which is a normal sized CC, and with the trained distance metric on every pair of neighboring CCs, a Minimal Spanning Tree (MST) is built. By cutting the edges, the end nodes of which belong to different text lines, CCs are grouped into different text lines. Unassigned
CCs are allocated applying similar post processing methods mentioned above.

There are always debates between top-down and bottom-up methods. Top-down methods may suffer from large curved documents, or multiple touching text lines, and bottom-up methods focus on local features, and many complicated computation and heuristics are needed. [8] presents a review of existing segmentation methods for handwritten text lines.

In this paper, we propose a method using seam carving to capture the global characteristics of the documents, which was first used in [9] for language-independent text line extraction. In order to capture more information in local regions, we constrain the scope and orientation of passing the energy, in order to transmit the energy of one point mainly to the neighbours in the same text line, and also avoid making the points accepting energy passed by the ones in the different text lines. Moreover, we extract all the text lines by computing the energy map only once, instead of recomputing the energy map after one text line is extracted, as in [9]. We also smooth the generated seams by polynomial fitting, in order to correct the sharp orientation changes along seams.

II. SEAM CARVING

Seam carving was first proposed for content-aware image resizing in [10]. For an image to be resized, an energy map is generated by the following energy function [10]:

$$e(I) = \left| \frac{\partial I}{\partial x} \right| + \left| \frac{\partial I}{\partial y} \right|$$  (1)

where $I$ is the image, and $\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial y}$ are the horizontal and vertical gradients respectively.

Seams in the horizontal and vertical orientations are extracted based on the energy map. Every seam is a connected path, all the pixels along which have lower energy values. Therefore, these extracted seams can be removed because they contain less information, if we want to shrink the image.

However, in order to extract text lines, we want to find the seams crossing more strokes and indicating where one text line probably appears. In [9], the energy map is calculated using Signed Distance Transform (SDT). In SDT, the pixels on the strokes have negative values, and the others in the background have positive values. So that, horizontal seams following local minima present the positions of the candidate text lines.

As shown in Fig. 1, the SDT of a binary image in Fig. 1(b) indicates that the nearer the points to the center axis of the strokes, the lower the values are in SDT. So that, in intra-space of consecutive words or text lines, the values are very high.

Assuming $E(I)$ is the energy map based on SDT, seams are generated using a minimum energy accumulator $M$, which is constructed as following [9]:

$$M(i, j) = 2 * E(i, j) + \min_{l=1}^{1}((M(i + l, j) - 1))$$  (2)

where we only consider continuous connected seams.

$$M accumulates minimum energy along every seam, from left to right, and the seams are generated in an inverse direction, from right to left, by choosing the minimum values in the right-most column in $M$. In [9], after each text line is extracted, the energy map need to be recomputed, this may cause large computation effort. In the next section, we will propose a method to detect all the text lines, by computing the energy map only once.

III. OUR PROPOSED METHOD

A. Preprocessing

The average height $AH$ and the average width $AW$ of CCs are first calculated for each document, and CCs are classified into three classes: small strokes, large components, and ordinary CCs, as described in [6]. Small strokes are mostly located relatively far from the central axis of the text lines, and large components are the CCs with long ascenders or descenders, either only belonging to one text line or connecting multiple text lines. These two types of CCs may cause the seams jumping between different text lines. In order to avoid unwanted disturbance, small strokes are removed, and for the large components, only the parts with high density values in the horizontal histogram are kept. For example, if a large component is detected, as shown in Fig. 2(a), we first include all the other strokes in its $3 \times AW$ forward and backward columns, as shown in Fig. 2(b). The corresponding horizontal projection histogram is shown in the right part of Fig. 3. As shown in Fig. 4, we smooth the histogram by convolving the histogram with a Gaussian kernel with mean $AH$, and derivative $AH/4$, and remove the parts of the large component with intensities lower than a threshold.

Unlike previous proposed methods, which always discard large components in the text line extraction process, we keep the main body parts. Because large components may be constructed by two long words in different text lines, due to their
long ascenders or descendents, if we just discard them, there will be a large gap between their neighboring CCs, so that the seams may easily jump to other text lines when they encounter these large gaps. Therefore, we keep main body parts of large components, not only avoiding large gaps, but also letting the main body parts contribute to the energy passing, positively.

B. Energy function

Distance maps are calculated separately for points inside the components and others in the background. For the points inside the components, we first extract contours of all components on the documents and calculate Euclidean distance transform, denoted as $C$. Only the values on the components are kept. So that, the points along the central axis of strokes have larger values. For the points in the background, we extract skeleton of components, and calculate the Euclidean distance transform, denoted as $S$. Only the values in the background are kept, so that, points far from the central axis of strokes have larger values.

In order to enhance the energy along the writing orientation of the text lines, we convolve $C$ by an ellipse-shaped Gaussian kernel, with major and minor axes of $3 \times AH$ and $H$ respectively. The Gaussian kernel is normalized by scaling all the values into $[0,1]$. We use multiple Gaussian kernels with different rotation angles for each pixel, and choose the one with the maximum energy value. Because applying Gaussian kernels, intra-space between two words in the same text line can accept energy from the components on the left and right, in order to make the energy flow along the writing orientation in the intra-space between components, and lead the seams to follow the writing orientation, instead of jumping among different text lines.

At last, we turn the sign of the positive values in $C$ to negative and assign the other zero values to the ones in $S$ at the same positions. The final signed distance transform is denoted as $E'(I)$.

C. Energy accumulation

Energy is accumulated from left to right, and the energy can be passed to all the points in the following columns on the right. In some cases, the intra-space of two text lines are very narrow, so that the components in one text line may accept the energy from different text lines. If the energy accepted from the same text line is lower, the seam along the text line with lower energy will jump to other text lines.

Moreover, we do not want the energy to be passed too far away. For example, if two text lines in one document are with different lengths and they are all aligned to the right. The pixels along the longer one can always accumulate more energy than those along the shorter one, and the components in the shorter text line can be easily affected by the larger energy in the longer text line, so that the seam may jump between these two text lines.

In order to weaken the effect of the energy passed by a point from too far away, we accumulate the energy by weighting based on the distances, indicating from how far away the energy is passed. We also set a maximum distance, so that the energy from the distance larger than the maximum distance will be discarded. Besides, for the new energy accumulation matrix $M'$, we generate $Hist$ for all the points in the column on the left of the column under consideration, recording all the energy accumulated so far and the distances where each energy is from. The longest length of $Hist$ is set to $\frac{1}{2}$ of the width of the document, and the elements in $Hist$ is first-in-first-out. If the size exceeds the limitation, the first added element will be discarded and the new element is added at the last position.

We initialize $M'(i,1)$ to $E'(i,1)$, and $Hist(:,1)$ to $E'(i,1)$. $M'$ is constructed as following:

$$dist = length(Hist(i-1,j-1)) : -1 : 1$$

$$e1 = Hist(i-1,j-1)./dist$$

$$e2 = Hist(i,j-1)./dist$$

$$e3 = Hist(i+1,j-1)./dist$$

$$M'(i,j) = \min(e1,e2,e3) + 2 \times E'(i,j)$$

where $dist$ is used to denote the distance for each energy, which has been already accumulated. For example, if the length of $Hist$ is $l$, the energy accumulated from the farthest distance is $l$ along the minimum energy accumulated path. The first element in $Hist$ is the farthest, and the last element is the nearest, namely the newest added one. So that, the energy in $Hist$ is normalized by dividing the corresponding distances, in order to weaken the effect of the energy from far away, and enhance the influence of the neighbouring energy. When we select the minimum one among $e1$, $e2$ and $e3$, for example $e1$ is selected, then $Hist(i,j) = Hist(i-1,j-1) \cup (i,j)$. After all the points in column $j$ are updated, the values stored in $Hist(i,j-1)$ can be discarded to save storage space.

As shown in Fig. 5, both $M$ and $M'$ are calculated based on the same distance map, however, in Fig. 5(a), we can see...
that from left, the energy is propagated with a nearly 90 degree flare angle facing to the right horizontally. So that the energy can be passed across different text lines. In Fig. 5(b), with our proposed method, the energy is constrained to be passed along the same text line, and avoid interfering with the neighboring text lines above and below.

**D. Seam extraction**

At the end of constructing $M'$, from every cell in the last column, we generate all seams from right to left and get a set of connected horizontal seams, denoted as Seams. If the height of the document is $H$, there will be $H$ seams in Seams. Fig. 6 shows the seams we found using normal seam carving and our proposed method. We can see that, in Fig. 6(a), the seams which are generated based on $M$, jump among different text lines, if we group the components touching the same seams together, we cannot get correct text lines, and many components are missing. This situation is caused by large intra-space between two words in the same text line. However, in Fig. 6(b), the seams with our proposed method are generated correctly, all of which are along the central axes of the components, even though the intra-space between some words are large.

**Fig. 6.** Seams generated by $M$ and $M'$ in Fig. 5. The red lines indicate the extracted seams.

According to Fig. 6(b), the seams only five 5 different values in the first column (the left most column) on the document, namely, the starting positions of candidate text lines. Therefore, in Seams, we group the seams with the same value in the first position into one set, denoted as $\text{Seams}_i$, $i \in [1, n]$, where $n$ is the number of candidate text lines. We only keep one seam in each set, which consists mostly smooth and similar writing orientation in any local areas. Therefore, we apply the polynomial curve fitting to every seam in one set, and choose the one with minimum distance to the fitted curve. The final seams we found are shown in Fig. 7, denoted as $s_i$, $i \in [1, n]$.

**E. Postprocessing**

After we generate all seams, for each $s_i$, we first put all the normal sized components, which only intersect with one $s_i$, into a component set $c_i$. For the remaining components, we will handle them in the following four cases separately:

1) Case 1: If a large component only intersects with one seam, then we just put them into the corresponding component set;

2) Case 2: If a large component does not intersect with any seams, we will assign them to the seam which is closest to its main body part, ignoring the long ascenders or descenders.

3) Case 3: If a large component intersects with multiple seams, we first thicken the intersected seams with height $AH$ as text regions, and check the percentage of foreground pixels of the main body part in the large component lying in each text region. If only one text region contains more than 70% of the foreground pixels, the large component should belong to this text region. If more than one text regions contain similar percentage of the foreground pixels, the large component should be split and assigned to the separate component sets. The split method we use was proposed in [6]. Fig. 8 shows an example of splitting large components, which across two text lines.

4) Small components: we assign the small components to the closest text lines.

**IV. Experiments**

**A. Evaluation method**

Let $I$ denote all the foreground pixels in one testing document, $G_j$ the set of pixels inside the $j$ ground truth region, and $R_i$ the set of pixels inside the $i$ results region. $T(s)$ is a function counting the set of pixels in $s$, and the matrix $\text{MatchScore}(i, j)$ is used to describe how the result region is matched to the ground truth:

$$\text{MatchScore}(i, j) = \frac{T(G_j \cap R_i \cap I)}{T((G_j \cup R_i) \cap I)}$$  \hspace{1cm} (8)
A result region is considered as a one-to-one match to the ground truth region, if the matching score is equal or above 95%. Assume \( N \) is the number of ground truth elements, \( M \) is the number of result elements, and \( o_2 \) is the number of one-to-one match pairs, then the detection rate (DR) and recognition accuracy (RA) are defined as follows:

\[
DR = \frac{o_2}{N} \tag{9}
\]
\[
RA = \frac{o_2}{M} \tag{10}
\]

Combining DR and RA, \( FM \) is the evaluation metric:

\[
DR = \frac{2 \times DR \times RA}{DR + RA} \tag{11}
\]

B. Experimental setup

We test our proposed method on ICDAR2013 Handwritten Segmentation Contest dataset [11]. In the testing dataset, there are 100 English and Greek (Latin language) documents, and another 50 Indian (Bengali) documents, totally 2649 text lines. Fig. 9 shows three portions of documents written in different languages. The handwritten documents contain large writing styles, multi-skewed text lines and touching connected components. For comparison, there are total 11 different algorithms, and many techniques are used, including run-length analysis, Gaussian filtering, energy minimization, histogram projection, connected components analysis, grouping method, seam carving, and etc. For details of different algorithms, please refer to [11].

C. Results

Fig. 10 shows the evaluation results of 13 different algorithms based on \( FM \), and our result has the label ‘NUS’ in the horizontal axis. The segmentation result of our proposed method is \( FM = 98.41 \), only 0.25% less than the best result, putting us in the second position. More results with details are shown in Table I. Most of our failure cases are mainly caused by small floating strokes and slitting large components. In Indian documents, some characters have different parts vertically, and the lower parts are sometimes misclassified to the lower text lines.

V. Conclusion

We propose a segmentation method for handwritten documents based on seam carving. However, unlike the previously proposed method which first used seam carving to extract text lines, we constrain the energy flowing directions, so that the energy can be mainly passed to neighbouring points in the same text lines, and jumping across different text lines can also be avoided. Moreover, by only calculating the energy map once, we can extract all the text lines, instead of recomputing after each text line is extracted.

In future work, we would like to improve our energy ac-
TABLE I. EVALUATION RESULTS

<table>
<thead>
<tr>
<th>Methods</th>
<th>M</th>
<th>M20</th>
<th>DR(%)</th>
<th>RA(%)</th>
<th>FM(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUBS</td>
<td>2675</td>
<td>2395</td>
<td>97.96</td>
<td>96.64</td>
<td>97.45</td>
</tr>
<tr>
<td>GOLESTAN-a</td>
<td>2646</td>
<td>2802</td>
<td>98.23</td>
<td>98.34</td>
<td>98.28</td>
</tr>
<tr>
<td>INMC</td>
<td>2650</td>
<td>2914</td>
<td>98.68</td>
<td>98.64</td>
<td>98.66</td>
</tr>
<tr>
<td>LRDE</td>
<td>2632</td>
<td>2568</td>
<td>96.94</td>
<td>97.57</td>
<td>97.25</td>
</tr>
<tr>
<td>MSHK</td>
<td>2696</td>
<td>2428</td>
<td>91.66</td>
<td>90.06</td>
<td>90.85</td>
</tr>
<tr>
<td>NUS</td>
<td>2645</td>
<td>2605</td>
<td>98.34</td>
<td>98.49</td>
<td>98.41</td>
</tr>
<tr>
<td>QATAR-a</td>
<td>2626</td>
<td>2404</td>
<td>90.75</td>
<td>91.15</td>
<td>91.15</td>
</tr>
<tr>
<td>QNAR-b</td>
<td>2609</td>
<td>2380</td>
<td>91.73</td>
<td>91.14</td>
<td>92.43</td>
</tr>
<tr>
<td>NSRCS(NoA)</td>
<td>2646</td>
<td>2477</td>
<td>92.37</td>
<td>92.48</td>
<td>92.43</td>
</tr>
<tr>
<td>ILSP(S0A)</td>
<td>2685</td>
<td>2546</td>
<td>96.11</td>
<td>94.82</td>
<td>95.46</td>
</tr>
<tr>
<td>TIE(S0A)</td>
<td>2675</td>
<td>2590</td>
<td>97.77</td>
<td>96.82</td>
<td>97.30</td>
</tr>
</tbody>
</table>

cumulation process to reduce the computation time. Moreover, we will improve the performance of splitting large components which touch multiple text lines, and we will also work on gray level documents, which have more challenges.

REFERENCES


(a) The segmented English document.

(b) The segmented Greek document.

(c) The segmented Indian document.

Fig. 11. Segmentation results for three handwritten documents, and each text line is marked as an unique color.