Arabic Handwritten Text Line Extraction by Applying an Adaptive Mask to Morphological Dilation

Muna Khayyat, Louisa Lam and Ching Y. Suen
Concordia University
Center for Pattern Recognition and Machine Intelligence
Montreal, Quebec H3G 1M8, Canada
m_khay, llam, suen@encs.concordia.ca

Fei Yin and Cheng-Lin Liu
Institute of Automation of Chinese Academy of Science
National Laboratory of Pattern Recognition
Beijing 100190, China
fyin, liucl@nlpr.ia.ac.cn

***Abstract*—This paper presents a robust method for handwritten text line extraction. We use morphological dilation with a dynamic adaptive mask for line extraction. Line separation occurs because of the repulsion and attraction between connected components. The characteristics of the Arabic script are considered to ensure a high performance of the algorithm. Our method is evaluated on the CENPARMI Arabic handwritten documents database which contains multi-skewed and touching lines. With a matching score of 0.95, our method achieved precision and recall rates of 96.3% and 96.7% respectively, which demonstrate the effectiveness of our approach.**

***Keywords*-Arabic script, Text Line Extraction, Adaptive Mask, Morphological Dilation, Smearing.**

I. Introduction

Text line extraction is one of the most crucial steps in structural analysis and preprocessing of handwritten documents, since it affects the performance of the document recognition system. Compared to printed documents, line extraction in handwritten documents is more challenging because of irregular spacing between lines, curved and multi-skewed lines, varying skew within the same line, and touching and overlapping lines.

Arabic handwritten script is naturally cursive, unconstrained and horizontal. This makes the extraction of Arabic handwritten lines challenging. Many script independent methods have been proposed in the literature. However, Arabic handwritten text lines extraction algorithms have either not been evaluated [4] or have reported higher error rates than other languages [3]. This is because Arabic script is more cursive than other scripts. Furthermore, the Arabic script consists of small connected components called diacritics. These diacratics are usually located above or below the major connected components of the scripts. In handwriting these diacritics are often located between the text lines. Accordingly, they are often misplaced in script independent text line extraction methods.

In this work, we propose a method for Arabic handwritten text line extraction using a dynamic adaptive mask. The mask keeps adapting to the document to find the best mask size and shape to separate the text lines. Moreover, this mask may have different shapes for different zones in the document. The power of this mask becomes apparent as it segments the document into big blobs that give the potential layout of the lines. The text within blobs repulse or attract depending on the characteristics of the Arabic script. Applying special techniques to disconnect touching lines as a preprocessing step is computationally expensive. Thus, we detect and separate touching components in an intermediate step within the algorithm, which is computationally more efficient. Our algorithm ensures affinity of the diacritics to their original lines.

The remainder of this paper is organized as follows: Section II discusses related work. Section III describes our method. Section IV presents the experimental results and the database used for testing our method. Finally, Section V concludes our paper.

II. Related Work

Numerous methods have been proposed to extract text lines from handwritten documents. These methods can be classified into the following six major categories: projection profile based, smearing methods, Hough transformation based, clustering or grouping methods, repulsive attractive methods that uses energy minimization systems, and stochastic methods which use stochastic learning algorithms.

Some of these methods deal with specific languages such as Chinese or Arabic scripts. Yin and Liu [1] presented a clustering method using Minimal Spanning Trees (MST) to extract lines from both Chinese and Latin-based documents. The results show that their method performs well on multi-skewed and curved text lines in handwritten documents. Kumar et al [2] proposed a graph based approach to extract text lines from Arabic unconstrained handwritten documents. Their approach is fast since it is based on connected components. However, it does not perform well in presence of touching components. Shi et al [8] extracted Arabic handwritten text lines by applying a direction filter; then, an adaptive thresholding algorithm was applied to adaptive local connectivity maps to form connected components. Finally, they extracted lines by grouping the connected components using a clustering algorithm.

Many script independent text line extraction algorithms have been proposed. Bukhari et al proposed [7] a script independent line extraction algorithm that uses ridges over smoothed images to estimate the central line of text lines parts. An active contour was applied over ridges to segment

978-0-7695-4661-2/12 $26.00 © 2012 IEEE
DOI 10.1109/DAS.2012.20

the lines. Li et al [3] proposed a script independent algorithm that uses the level set for line segmentation. These two approaches perform well on Arabic handwritten documents. Nevertheless, they suffer from the high computational cost.

Ziaratban and Faez [4] used a bottom up algorithm that segments the document into adaptive blocks; then, the skew of each block is estimated. Three parameters were defined to adapt the method to different writers. Different techniques were combined to produce better results or to adapt to scripts of special characteristics. Ouwayed et al [5] implemented a text extraction system using various local techniques including snakes (Repulsive Attractive Methods) to create a contour that segments the lines into local zones. Then, the orientation of each zone was detected using special projection profile histograms.

III. Proposed Method

Arabic handwritten script is a horizontal cursive script by nature. Based on this fact, we used a horizontal dynamic mask to perform appropriate smearing to separate Arabic text lines in a document. Algorithm 1 presents the flowchart of our method. The method first identifies the characteristics of the document and its connected components to set the parameters and thresholds of the algorithm. The final smearing of the document is decided by the dynamic mask. The recursive function $separateLines(blob)$ plays an important role in our method, as it breaks up blobs according to the attraction and repulsion of the text within those blobs. Finally, diacritics and disjoint horizontal lines are merged to form the text lines. Figure 1 illustrates the major steps of our method.

A. Document Analysis and Preprocessing

Before applying the line segmentation algorithm the document is preprocessed and then analyzed, which allows the algorithm to learn the properties of the document, and tune the parameters and thresholds, of the line segmentation algorithm.

A 3×3 Gaussian filter is applied to the document to remove the noise. Then, Otsu binarization algorithm [6] is applied to the document.

The average height and width of the major connected components in the document are calculated. These parameters are needed to initialize the width of the dynamic mask wd. The horizontal projection profile $f(y, p(y))$ of the document is found, which reflects the nature of the document and the distribution of the text lines. From $f(y, p(y))$ the algorithm considers the significant peaks and valleys and finds the slope between each peak and its neighboring valley. The slopes reflect the skew of the lines in different zones, and these slopes would determine the slope of the smearing mask within different zones.

Figure 2 shows the horizontal projection profile of two different documents. The horizontal projection profile in Figure 2 (a) shows that the lines are not well separated and words are sparse all over the document. In addition, there are no deep valleys in many parts of the profile, which give an indication that the lines are skewed and close to each other. However, the profile of Figure 2 (b) shows that the lines in the documents are nicely separated, since it has deep valleys and almost uniform peaks and valleys. The slopes between peaks and valleys of the horizontal projection profile are calculated using the following equation:

$$slope_{prof} = tan(\theta) = \frac{p(y_{peak}) - p(y_{valley})}{y_{peak} - y_{valley}} \quad (1)$$

where $p(y)$ is the number of white pixels in row y, while y_{peak} and y_{valley} are the coordinates of the lines where the peak and the valley are located.

Algorithm 1 Line Segmentation

$h_{avgc} \leftarrow$ Average height of major connected components
$w_{avgc} \leftarrow$ Average width of major connected components
$wd \leftarrow \frac{{w_{avgc}}^2}{h_{avgc}}$
repeat
  $blobs \leftarrow smearDocument(wd)$
  for all $blob_i \in blobs$ **do**
    if $h_{blob} > h_{blob-threshold}$ **then**
      $separateLines(blob)$
    end if
  end for
  $wd \leftarrow wd - 3$
until $\forall blob_i \in blobs$ $(h_{blob} \leq h_{blob-threshold})$
blobs re-labeling
for all $blob_i \in blobs$ **do**
  if $blob_i$ is small **then**
    $mergeBlobs(blob_i, blob_j)$ where $blob_j$ is the closest big blob to $blob_i$
  end if
  if $checkKaf(blob_i)$ **then**
    $mergeBlobs(blob_i, blob_j)$ where $blob_j$ is the closest big blob to $blob_i$
  end if
  if $checkDiacritics(blob_i)$ **then**
    $mergeBlobs(blob_i, blob_j)$ where $blob_j$ is the closest big blob to $blob_i$
  end if
end for
for all $blob_i \in blobs$ **do**
  $doHorizontal(blob_i)$
end for

B. Morphological Dilation and Dynamic Mask

The binary document is smeared to produce big connected components (blobs) as shown in Figure 1 (b). Document A is dilated using a dynamic mask (structuring element) B to

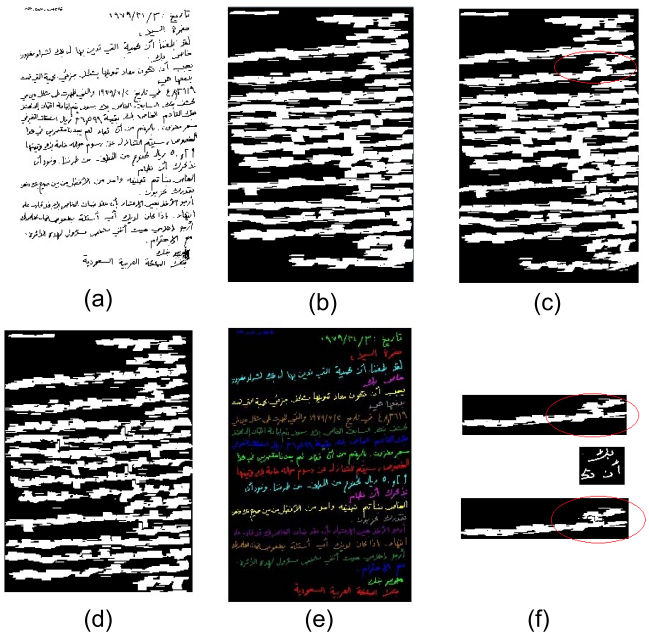

Figure 1: Steps of the line segmentation algorithm: (a) Preprocessed document, (b) Initial smearing, (c) Dynamic adaptive smearing, (d) Final smeared blobs, (e) Resulted lines, and (f) Illustration of line separation.

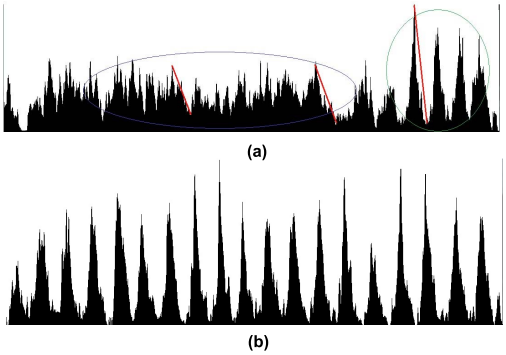

Figure 2: (a) The horizontal projection profile of a document, where words are sparse and lines are of different skews, (b) The horizontal projection profile of a document where lines are well separated.

produce a new smeared document S as follows:

$$S = A \oplus B = \{z|[(\hat{B})_z \cap A] \subseteq A\} \quad (2)$$

where $\hat{B} = \{w|w = -b, b \in B\}$, and A, B and S are sets in Z^2.

Initially, the document is dilated with a binary mask B of only 1's with height h_m and width w_m.

$$h_m = 1 \quad (3)$$

$$w_m = \frac{{w_{avgc}}^2}{h_{avgc}} \quad (4)$$

where w_{avgc} and h_{avgc} are the average width and height of the Pieces of Arabic Words (PAW) of the document respectively.

The blobs of the smeared documents are analyzed to perform the suitable smearing to the document. If any blob has a height greater than $4 \times h_{avgb}$, where h_{avgb} is the average height of the potentially smeared document blobs, then the slopes between the peaks and valleys within a zone where the blob is located are calculated using Equation (1). Thus, if the absolute value of the calculated slope $tan(\theta)$ is greater than a predefined threshold (3.5), then the height of the mask h_m changes to a maximum of 3 and the width of the mask decreases by 3.0 accordingly. The mask for that zone will be a slanted line, with a slope equal to that of the lowest blob in the zone. The blob's slope is computed using the mean square method. Finally, the zone of that blob is re-smeared with the new mask. This step is repeated for all blobs until the document is smeared. Figure 1 shows the potentially smeared document (b) and the smeared document after dynamically changing the shape, dimensions and inclination of the mask for each zone (c).

C. Splitting Lines

The smeared document is segmented into big connected components (blobs). The actual height of a blob, which is the maximum number of white pixels in one of the columns (see Figure 3), is found. If the height is greater than $2.5 \times h_{avgc}$, then the blob is passed to the recursive function $separateLines(blob)$ for line separation. The threshold depends on the average height of the connected components of the original document h_{avgc}, which is the average height of the PAW, and the average depth of the valleys in that blob. The line separation function $separateLines(blob)$

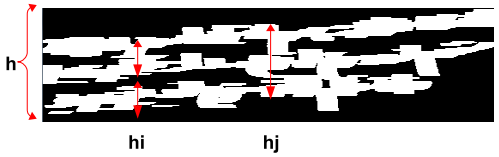

Figure 3: The bracket on the left represents the image height. The maximum number of white pixels in a column represents the actual height of the blob.

looks for the point (x, y) where, x is the column with the maximum run of white pixels, while y is the row with the minimum horizontal projection around x. A block of size $3 * h_{avgc} \times 3 * w_{avgc}$ centralized at (x, y) is taken, and the connected components of the text in this block are extracted. The connected components closer to the upper bound of the given block are attracted to each other, and the ones below them are repulsed. This occurs by removing the pixels below the lower profiles of the upper connected components. We follow this attraction and repulsion criterion, since

the Arabic script is a cursive horizontal script, in which major connected components are attracted horizontally and repulsed vertically. For touching components, if the component has a height greater than 1.8 of the average height, then it is most probably two touching PAWs, and the separation will occur in the thinnest part in the middle part of the component. This can be illustrated in Figure 4 and Figure 1 (f). The function keeps iterating recursively until the width of the blob is less than a predefined threshold. To avoid infinite loops or bad splitting, if the function iterates more than 7 times, the function stops iterating and returns a negative flag. The width of the mask is dynamically reduced by 3 pixels and the document is re-smeared accordingly.

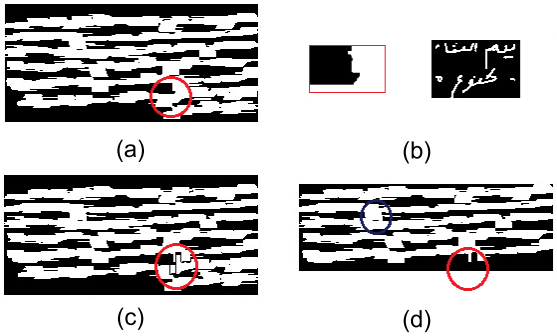

Figure 4: Line Separation Algorithm: (a) The first detected blob, since its actual height is above the threshold. The red circle in the blob shows the block with the maximum contiguous white pixels. (b) The text within the candidate block and the projection of this block are shown. (c) The red circle shows the splitting of the blob. (d) The red circle shows the blobs are seperated. The blue circle shows the second candidate block to be separated.

After performing the line separation, the number of connected components will increase, because many blobs are separated into two or more blobs. As a result, the algorithm extracts the connected components again and re-labels them (blobs).

D. Diacritics Affinity and Merging Horizontal Lines

In Arabic script diacritics and dots are small connected components that are located above or below the words. Thus, they are often located between the text lines. The diacritics are sometimes not merged within the big blobs because of their location. This results in having small blobs with a small height. The affinity of these diacritics will grow between the nearest big blobs, the Euclidian distance between the diacritics blobs and the big blobs is used to find the nearest blob. Some diacritics are relatively wide, such as the diacritics of the "Kaf" in Arabic as shown in Figure 5. These diacritics may not be merged in this step. Accordingly, after passing this step the algorithm looks for the connected components of relatively sizeable width and distinguish between them and the PAW. "Kaf" is always located above the words, so even if it appears closer to the upper blob, it is always merged to the blob below it. By the end of this step all diacritics will be merged to the suitable line.

Some blobs from the same line may not touch because of the width of the mask. After merging the diacritics and separating the lines, the algorithm looks for blobs with left and right ends in the same horizontal region. Those blobs belong to the same line. As a result, the algorithm will connect them horizontally and group them in one line in which they will attract horizontally.

IV. Experimental Results

Our algorithm was tested on the CENPARMI database (Section IV-A), which contains touching and unconstrained lines. Our algorithm was able to well separate these touching components and interfering lines. Figure 5 shows the results of separating touching and interfering lines using our method. Furthermore, the results show that our algorithm outperformed the well known MST algorithm [1] that is based on connected components and it performed very well on Chinese and Latin-based scripts.

Figure 5: Line separation using our method. First row shows some touching components and second row shows the interfering lines. Circles surround the diactarics of the Kaf.

A. Database

We evaluated the performance of our method using the CENPARMI Arabic unconstrained handwritten documents database. This database consists of 146 Arabic handwritten documents for a total of 2,137 lines written by different writers. The documents contain multi-skewed and touching lines, and were digitized with a resolution of 300dpi.

B. Evaluation

We used the $precision$, $recall$ and $f1_{score}$ metrics to evaluate our method. An $M \times N$ confusion matrix is found between the M ground truth lines and the N result lines. Given that g_i is a ground truth line, r_i is a result line, $P(x)$ is a black pixel in the line x, and $T(x)$ counts the number of pixels in zone x, the matching score (MS) between the result and ground truth documents is computed as follows:

$$MS(r_i, g_i) = \frac{T(P(r_i) \cap P(g_i))}{T(P(r_i) \cup P(g_i))} \quad (5)$$

The confusion matrix was filled with the MS scores between lines. For each result line, if the score is above a predefined threshold then the line is considered as true positive TP. Result lines that are not matched are considered false positive FP. Finally, ground truth lines that are not matched are considered as false negative FN. The $precision$, $recall$ and $f1_{score}$ are computed as follows:

$$precision = \frac{TP}{TP + FP} \tag{6}$$

$$recall = \frac{TP}{TP + FN} \tag{7}$$

and

$$f1_{score} = \frac{2 \times precision \times recall}{precision + recall}. \tag{8}$$

Table I shows the $precision$, $recall$ and $f1_{score}$ of our algorithm and the MST algorithm for line segmentation [1]. The results show that our method outperformed the MST algorithm.

Method	$Precision$	$Recall$	$F1_{score}$
MS	0.95		
Our Method	0.96319	0.967228	0.965421
MST	0.816784	0.871951	0.843466
MS	0.90		
Our Method	0.975746	0.979859	0.977799
MST	0.84051	0.89728	0.867967

Table I: Experimental Results on CENPARMI Arabic Handwritten Documents Database

Table II compares our method with Kumar et al [2] method using the handwritten Arabic proximity database [9] with $MS = 0.95$. The methods produce similar results. However, the adaptive mask in our method introduces a new technique to identify a potential layout of the handwritten text lines. The results of our algorithm can be significantly improved by applying state of the art methods to disconnect touching components, and to separate the blobs from the critical regions detected by our algorithm. Moreover, training on some Arabic handwritten documents to set the thresholds can be expected improve the results.

Method	$Precision$	$Recall$	$F1_{score}$
Our Method	0.90309	0.91536	0.909185
Kumar et al [2]	0.9161	0.9017	0.909

Table II: Experimental Results on Database [9] with $MS = 0.95$

V. Conclusion

We proposed a robust Arabic handwritten text line extraction algorithm that uses a dynamic mask and is based on document smearing. Smearing usually cannot deal well with overlapping and touching lines. However, our dynamic mask and line splitting criterion that depends on the attraction and repulsion of the connected components, overcame the aforementioned drawback and made the algorithm robust to touching and overlapping lines.

Moreover, our algorithm introduces a new way to identify a potential layout of the text lines and detect the critical regions to break up text into lines. Thus, different techniques can be proposed for text repulsion and attraction at these regions to improve the text line segmentation results.

Acknowledgment

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada, Concordia University, and the National Natural Science Foundation of China (NSFC) under grant 60825301 and grant 60933010.

References

[1] Fei Yin and Cheng-Lin Liu, "Handwritten Text Line Extraction Based on Minnimum Spanning Tree Clustering," *Pattern Recognition*, vol. 42, issue 12, pp. 3169-3183, 2009.

[2] Jayant Kumar, Le Kang, David Doermann and Wael Abd-Almageed, "Handwritten Arabic Text Line Segmentation Using Affinity Propagation," *Document Analyis Systems*, pp. 135-142, 2010.

[3] Li Yi, Yefeng Zheng, David Doermann, and Stefan Jaege, "Script-Independent Text Line Segmentation in Freestyle Handwritten Documents," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, val. 30, no. 8, pp. 1313-1329, 2008.

[4] Majid Ziaratban and Karim Faez, "An Adaptive Script-Independent Block-Based Text Line Extraction," *In Procceedings of the 20th International Conference on Pattern recognition (ICPR '10)*, pp. 249-252, 2010.

[5] Nazih Ouwayed, Abdel Bela, and Francois Auger, "General Text Line Extraction Approach based on Locally Orientation Estimation," *In Procceedings of the 17th Document Recognition and Retrievall Conference (DRR 2010)*, 2010.

[6] Nobuyuki Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Sys., Man., Cyber*, vol. 9, issue 1, pp. 6266, 1979.

[7] Syed Saqib Bukhari, Faisal Shafait and Thomas M. Breuel, "Script-Independent Handwritten Textlines Segmentation using Active Contours," *In Preceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR2009)*, vol. 2, pp. 446-450, 2009.

[8] Zhixin Shi, Srirangaraj Setlur, and Venu Govindaraju,"A Steerable Directional Local Profile Technique for Extraction of Handwritten Arabic Text Lines," *In Preceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR2009)*, vol. 1, pp. 176-180, 2009.

[9] Handwritten Arabic Proximity Database. Language and Media Processing Laboratory. http://lampsrv02.umiacs.umd.edu/projdb/project.php