Multi-Oriented and Multi-Sized Touching Character Segmentation using Dynamic Programming

Partha Pratim Roy
Computer Vision Center
UAB, Barcelona, Spain
partha@cvc.uab.es

Umapada Pal
CVPR Unit, ISI
Kolkata, India
umapada@isical.ac.in

Josep Lladós
Computer Vision Center
UAB, Barcelona, Spain
josep@cvc.uab.es

Mathieu Delalandre
Computer Vision Center
UAB, Barcelona, Spain
mathieu@cvc.uab.es

Abstract

In this paper, we present a scheme towards the segmentation of English multi-oriented touching strings into individual characters. When two or more characters touch, they generate a big cavity region at the background portion. Using Convex Hull information, we use these background information to find some initial points to segment a touching string into possible primitive segments (a primitive segment consists of a single character or a part of a character). Next these primitive segments are merged to get optimum segmentation and dynamic programming is applied using total likelihood of characters as the objective function. SVM classifier is used to find the likelihood of a character. To consider multi-oriented touching strings the features used in the SVM are invariant to character orientation. Circular ring and convex hull ring based approach has been used along with angular information of the contour pixels of the character to make the feature rotation invariant. From the experiment, we obtained encouraging results.

1. Introduction

In artistic or graphical documents the text lines appear frequently in different orientations other than the usual horizontal directions. The OCR systems available commercially do not perform well in handling them. The main difficulty arises from the severely merged or degraded characters. Incorrect segmentation of merged/touching characters is still one of the main causes for recognition errors. It is difficult for segmentation and recognition of documents containing multi-oriented text. We show an example in Fig.1 to illustrate the problem. It can be seen, the word "PLANET" contains a touching string "LANE".

There are many published papers towards the recognition and segmentation of the touching characters of normal horizontal direction [2, 12]. Touching character recognition methods can be divided into two classes: segmentation based approach and holistic approach [12]. In segmentation based recognition domain, at first touching string is

segmented into characters and then segmented characters are passed for recognition [1, 7, 8]. In holistic methods, the algorithms use the word as a single entity, and try to recognize it using features of the word [6, 9]. Recently, we have proposed a method based on convex hull approach [10] for segmenting two-touching characters in multi-size and multi-oriented environment.
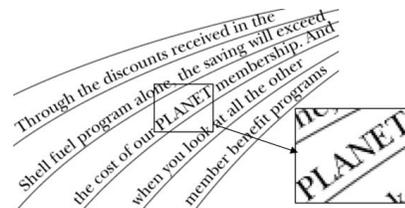

Figure 1. Example of an advertisement shows orientation of text characters.

To handle multi-oriented touching strings within graphical documents, in this paper, we propose a segmentation scheme to recognize n-character touching patterns in arbitrary orientations. When two or more characters touch, they generate a big cavity region at the background portion. We used this background information to find the segmentation point in our method. To handle the background information convex hull is used. In the proposed scheme, at first, a set of initial segmentation points is predicted based on the concave residues of the convex hull of a touching string. Next, based on the initial segmentation points, the touching string is segmented into primitive segments. A primitive segment consists of a single character or a sub-image of a single character. Next these primitive segments are merged to get optimum segmentation and dynamic programming is applied using total likelihood of characters as the objective function. SVM classifier is used to find the likelihood of a character. To consider multi-oriented touching strings the features used in the SVM are invariant to character orientation. Circular ring and convex hull ring based approach has been used along with angular information of the contour pixels of the character to make the feature rotation invariant.

The rest of the paper is organized as follows. In Section 2, we explain our feature extraction procedure to handle multi-oriented touching strings. Proposed segmentation approach is discussed in Section 3. The experimental results are discussed in Section 4. Finally conclusion is given in Section 5.

2. Character Feature Extraction and Model Generation

Recognition of individual characters in multi-oriented and multi-sized environment drives the segmentation hypothesis of n-touching characters in our system. Let us describe the feature extraction and character model generation method as follows.

Feature Extraction: We proposed a zone-based signature for character recognition in our earlier paper [11]. Circular ring and convex hull ring based concept have been used to divide a character into several zones to compute features. To make the system rotation invariant, the features are mainly based on angular information of the external and internal contour pixels of the characters. Given a sequence of contour pixels, the change of angle of each pixel is calculated from the neighbour pixels. The angles obtained from all the contour pixels of a character are grouped into 8 bins corresponding to eight angular intervals of $45°$. For a character, frequency of the angles of 8 bins will be similar even if the character is rotated at any angle in any direction. To take care of broken characters, the feature has been modified and described below.

Circular and convex hull rings are constructed on a character as follows. A set of circular rings is defined as the concentric circles considering their center as the center of minimum enclosing circle (MEC) of the character and the minimum enclosing circle is the outer ring of the set. Similarly, convex hull rings are also constructed from the convex hull shape of the character. The radii of the rings are in arithmetic progression. In Fig.2 we show 4 circular and convex hull rings on two characters. MEC is circular in nature for all objects. Whereas, convex hull of an object is a closed polygon and its shape differs depending on the shape of objects. Convex hull ring may not be circular and it is better adjusted to the shape of the object.
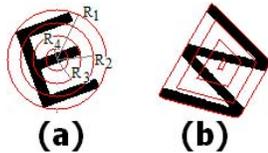

Figure 2. (a) Circular (b) Convex hull rings.

To get more local feature, we compute angular information (slope) of the contour pixels with respect to the center of the MEC. We grouped the contour pixels of a character into 8 bins based on their angular information with respect to the neighbor pixels. Now, angular information (slope) with respect to the centre of MEC of the bin-wise contour pixels is computed. Here by the slope of a pixel V_i with respect to centre of MEC, we mean the angle formed at V_i by $\overrightarrow{OV_i}$ and $\overrightarrow{V_iV_{i+k}}$ (where O is the center of the MEC of a character and k is distance of neighbor pixels as discussed in [11]). The angle obtained from all the contour pixels of a bin are grouped into 8 sets corresponding to 8 angular information of $45°$ (=360/8). Thus, for 8 bins of contour pixels, we have $8×8 = 64$ dimensional slope features.

Finally, considering 7 circular rings and 7 convex hull rings, we have 56 ($8×7$) feature from convex hull ring, 56 ($8×7$) features from circular ring and 64 ($8×8$) features from angular information with respect to center of MEC. As a result, we have 176 (56+56+64) dimensional feature vector for the classification. This feature has been selected based on experiment. Normalization of the feature is done to obtain scale invariance [11].

Character Shape Model: Support Vector Machine (SVM) has been used to build the character shape model from corresponding feature of our training data. Both English uppercase and lowercase alpha-numeric characters were considered for our experiment, so we should have 62 classes (26 for uppercase, 26 for lowercase and 10 for digit). But because of shape similarity of some characters/digits, we group them in 40 classes. Given a set of primitive segments, we compute the recognition confidence to obtain the corresponding class and use this value as the cost function in our dynamic programming approach.

3. Proposed Segmentation approach

There may exist touching or non-touching characters in a word. Before passing a component into our segmentation approach, we detect if the component is touching or isolated. We apply a Connected Component (CC) labeling to the word image and extract individual components. For each component, we compute the recognition confidence for all character class models using our SVM classification process and rank the confidence scores in descending order. If we recognize a component with a very high accuracy, we assign it as a non-touching character. If the difference between top two recognition scores of a component is high, it is also considered as non-touching character. The rest of the components are considered as touching and we process them for segmentation. If a word is rotated to a certain angle, we measure the angle from the minimum rectangular bounding box of the word shape. We compute the bounding box of the word and find the angle (α) of the major axis. The more are the characters in a word, the better is the approximation of the angle. An approximate height of the word (H_w) is found from its bounding box. It is to be noted that, when there are few characters in the word and it

includes characters having ascendants and descendants (for e.g. 'y', 'l' etc.), α gives only an idea of the inclination of the shape. In our experiment α is used to arrange the primitive segments in an order. Details of the segmentation method are discussed below.

3.1. Initial Segmentation Point Detection

For a touching component, we find the concave residues using convex hull property [10]. These residues are regarded as the segmentation zones. We assume, when two or more characters touch, they generate a big cavity region at the background portion. The residues found from convex hull of a touching character are shown in Fig.3. In our process, the residues having height more than stroke-width are considered. The stroke-width (St_w) of the word is the statistical mode of object pixels' run lengths [10]. We find initial points in these segmentation zones to segment a touching string into possible primitive segments.

We apply a standard polygonal approximation method due to Douglas and Peucker [4], to the residue border contour pixels. This is done to find the contour pixels where it lacks the smoothness. The line of residue surface level is regarded as the initial rough estimation of the polyline. Using this crude initial guess, the other vertices are approximated using a tolerance threshold greater than St_w. After approximation, we will have the list of vertices which are treated here as key-points. The advantage of using polygonal approximation is that, it helps us to provide the key-points which are at the corner of edges in that segmentation zone. These key points information is necessary for touching character segmentation because, usually when the characters touch, they form a corner at the touching region. Further, we exclude the key-points which are very close to residue surface level. This is because, we assume the touching position between characters is closed to deepest point from that residue surface level. The remaining key-points are the initial segmentation points. In Fig.4(b), we have shown the initial segmentation points for the image Fig.4(a).
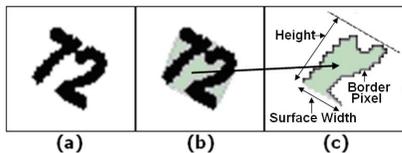

Figure 3. (a) Touching character. (b) The residues of the convex hull are shown. (c) Different parameters of the residue.

3.2. Primitive Segmentation

For each initial segmentation point, we find a segmentation line passing through this point. From a segmentation point, we compute a line which is perpendicular to the corresponding residue surface label and that line segments the

image. The line is tuned more by checking the neighborhood contour pixels in which the length of the segmentation line is small. Now, for each initial segmentation point, we find the corresponding segmentation line. If we have n segmentation lines, the image is over-segmented in n+1 sub-images. We remove some segmentation lines which are very closed and larger than half of the width of the word. The (candidate) segmentation lines of Fig.4(a) have been shown in Fig.4(c). Note that, there were 7 initial segmentation points and we have got 5 segmentation lines. These segmentation lines split the touching component into primitive segments. Now, we will merge some primitive segments for correct segmentation and we will use dynamic programming technique for the purpose.
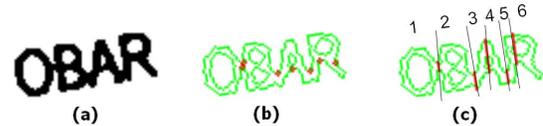

Figure 4. (a) Touching string. (b) Initial segmentation points found from concave residues. (c) Candidate segmentation lines obtained from selected segmentation points.

3.3. Merging of Primitive Segments by Dynamic Programming

The core of a dynamic programming (DP) algorithm [5] is the module that takes a set of symbols (list of primitive segments in our case) and a set of labels (possible characters) and returns optimum assignment of labels to symbols assuming that an optimum assignment is the sum of the sub-assignments. Given a touching image, the primitive segments are merged so that the average character likelihood is maximized using DP. The likelihood of each character is calculated using a recognition accuracy obtained by SVM. The number of primitives is usually 1.2 to 2 times as many as the number of actual characters in the touching character in our experiment.

To apply the DP, the primitive segments are aligned in horizontal direction by angle α (discussed above). They are sorted from left to right according to the location of their left-most pixel. Let $S_1, S_2, ..S_n$ are list of n primitives. In Fig.4(c) the primitive segments are indexed according to their position. We use two tables to store the character likelihood of primitive segments after merging (see Table1). In the Score Table ST, we enter the classification score $\{c_{uv}\}$ and in the Label Table LT, we enter the character classification label $\{l_{uv}\}$, where $1 \leq u \leq v \leq n$.

$$c_{uv} = \texttt{score of} \bigcup_{i=u}^{v} S_i \texttt{ and } l_{uv} = \texttt{label of} \bigcup_{i=u}^{v} S_i$$

In these tables, the cells correspond to recognition result of cumulative grouping of primitive segments. The possible merging result of primitive segments of Fig.4(c) are shown

13

in Table1(a) and Table1(b). For e.g., in Table1(b), the cell $l_{3,5}$ indicates the character likelihood of merging primitive segments s3, s4 and s5. And the result obtained by our SVM is 'm'. If the classification score of merged segments is less than a threshold (empirically decided), we do not consider it. Cumulation of primitive segments is continued till the width of the resultant image is less than $1.2 \times H_w$.

(a) — Score Table ST

	S1	S2	S3	S4	S5	S6
S1	0.969					
S2		0.856				
S3		0.705	0.566			
S4			0.380			
S5			0.183	0.645	0.158	
S6				0.205	0.839	0.103

(b) — Label Table LT

	S1	S2	S3	S4	S5	S6
S1	O					
S2		B				
S3		a	t			
S4			A			
S5			m	n	l	
S6				m	R	2

Table 1. (a) Score Table ST and (b) Label Table LT of character string "OBAR"

Next, we check the total likelihood of the character groups. The group having maximum likelihood is chosen and the corresponding primitive merging are their segmentation result. In Fig.4(c), 1st segment corresponds to letter 'O', 2nd segment corresponds to letter 'B', 3rd and 4th segments correspond to letter 'A' and 5th and 6th segments correspond to letter 'R'. The assignment for the characters $O \; B \; A \; R$ is also represented by :
$i \rightarrow 1 \; 2 \; 3 \; 4$ and $j(i) \rightarrow 1 \; 2 \; 4 \; 6$
where i denotes the letter number, j(i) denotes the number of the last primitive corresponding to the i-th letter. Note that the number of the first primitive segment corresponding to the i-th letter is j(i-1)+1. Given j(i), (i = 1,..n), the total likelihood of characters is represented by

$$L = \sum_{i=1}^{n} l(i, j(i-1) + 1, j(i)) \qquad (1)$$

where $l(i, j(i-1) + 1, j(i))$ is the likelihood for i-th letter. The optimal assignment (the optimal segmentation) that maximizes the total likelihood is found in terms of the dynamic programming as follows. The optimal assignment $j(n)^*$ for n-th letter is the one such that

$$L_{j(n)}^* = L(n, j(n)^*) = \text{Max} L(n, j(n)) \qquad (2)$$

where $L(k, j(k))$ is the maximum likelihood of partial solutions given $j(k)$ for the k-th letter, which is defined and calculated recursively by
$L(k, j(k)) = \text{Max}_{j(1), j(2)..j(k-1)} \sum_{i=1}^{k} l(i, j(i-1)+1, j(i))$

$= \text{Max}_{j(k-1)}[l(k, j(k-1)+1, j(k)) + L(k-1, j(k-1))]$ (3)

and $L(0, j(0)) = 0$ for $j(0) = 1, 2, ...m$ (4)

Starting from (4), all $L(k, j(k))$'s are calculated for $k = 1, 2, ..., n$ using (3) to find $j(n)^*$ using ((2)). The rest of $j(k)^*$'s $(k = n-1, n-2, ..., 1)$ are found by back tracking a pointer array representing the optimal $j(k-1)^*$'s which maximizes $L(k, j(k))$ in (3).

Given a segment group, feature vector is calculated for a character class. Based on the character likelihood, total likelihood of a word is found in terms of the dynamic programming technique. In Fig.5 we have shown the final segmentation result of the touching character of Fig.4(a).

Figure 5. Final segmentation lines are drawn

4. Data Collection and Experimental Result

To the best of our knowledge, there exists no standard database to evaluate character segmentation methods in a multi-orientated and multi-size context. For our experiments we have constituted our own database using real as well as synthetic data. The real data is collected from maps, newspapers and magazines. They are digitized in grey tone with 300 dpi and we have used a histogram based global binarization algorithm.

The synthetic data have been generated using the system described in [3]. This database is put online[1] for the use of other researchers. It is composed of single-word images with different scales, orientations and fonts, with corresponding groundtruth at character level. The words are selected from a dictionary (of 52 country names), with random scaling and rotation parameters and using predefined fonts. Each word consists of 7-8 characters in average. In each word, this data generation method looks for the pairs of successive characters, and makes them connected according to a Boolean value. Overlapping between characters is controlled using a Gaussian function.

To check the capability of the features towards isolated character recognition, the proposed features has been tested using SVM classifier in database of graphical data [11] containing isolated characters. We have obtained 98.44% recognition accuracy by 5-fold cross validation.

In our experiment on touching character segmentation, we tested our scheme on 380 words (150 real and 230 synthetic). The synthetic data contains touching as well as non-touching characters. There were 1790 characters touched in 815 components out of these 380 words. The touching strings are of different size and orientation. We obtained 91.44% accuracy in overall experiment. To get an idea about the segmentation results, in Fig.6, we have shown some touching images with their segmentation results. Fig.7 shows some wrong segmentation of touching characters from our method. We provide the accuracy of touching characters based on number of characters touched in that component in Table 2. We noted that, our system provides better results on 2-character touching strings than 3 or more character touching string.

---

[1]http://mathieu.delalandre.free.fr/projects/sesyd/charseg.html

14

We compare our results with existing system on normal touching of horizontal direction in Table 3. Though the accuracy is less compared to that of result obtained by [12, 13], but we think for multi-oriented environment it is a good performance.

Table 2. Segmentation results

Number of character	Accuracy
2	92.18%
3	90.40%
≥ 4	88.60%

Table 3. Comparison of segmentation results

Method	Touching Direction	Accuracy
Song et al. [12]	Horizontal	98.3%
Chen and Wang [13]	Horizontal	96.0%
Our System	Multi Rotation	91.44%

It is noted that most of the segmentation errors are for the following cases. (a) When a touching string can be segmented in more than two ways to get the valid segmented characters. For example, in Fig.7(a) the touching string was formed from the characters 'r' and 'm'. But this touching string can be visualized as 'n' and 'n' also, and our system segmented this string into 'n' and 'n' instead of 'r' and 'm', which we consider as erroneous. (b) The character shapes like 'h' (Fig.7(b)) may be split in two parts and our system segments this character into 't', 'I'. We also considered it as wrong segmentation. (c) Since our method is based on convex hull, when touching is made in two or more positions, then we may not find any segmentation point in the touching cavity region. Hence we get erroneous results.

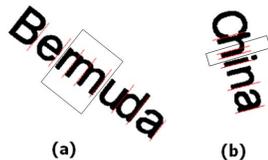**Figure 6. Correct segmentation results**

(a)   (b)

Figure 7. Wrong segmentation results

5. Conclusion

In this paper, we have proposed a scheme towards segmentation of multi-oriented and multi-sized touching strings. The algorithm works on segmenting the touching character into primitive segments and then finds the best sequence of model characters shapes based on Dynamic Programming using these primitive segments. This algorithm is efficient to take care of characters string in noisy environment. To the best of our knowledge, this work is pioneer towards multi-oriented and multi-sized touching strings segmentation. Certainly, the efficiency can be improved by using dictionary of words in dynamic programming and context information.

6. Acknowledgements

References

[1] N. Arica and F. T. Yarman-Vural. Optical character recognition for cursive handwriting. *IEEE Transactions on PAMI*, 24(6):801–813, 2002.

[2] T. C. Chang and S. Y. Chen. Character segmentation using convex-hull techniques. *IJPRAI*, 13(6):833–858, 1999.

[3] M. Delalandre, T. Pridmore, E. Valveny, E. Trupin, and H. Locteau. Building synthetic graphical documents for performance evaluation. *GREC*, pages 288–298, 2008.

[4] D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10(2):112–122, 1973.

[5] F. Kimura, S. Tsuruoka, Y.Miyake, and M. Shridhar. A lexicon directed algorithm for recognition of unconstrained handwritten words. *ICICE Trans. on Information and Systems*, E77:785–793, 1994.

[6] S.-W. Lee and Y.-J. Kim. A new type of recurrent neural network for handwritten character recognition. *ICDAR*, 01:38, 1995.

[7] S. Liang, M. Shridhar, and M. Ahmadi. Segmentation of touching characters in printed document recognition. *Pattern Recognition*, 27(6):825–840, 1994.

[8] Y. Lu. On the segmentation of touching characters. *ICDAR*, 1:440–443, 1993.

[9] J. Rocha and T. Pavlidis. Character recognition without segmentation. *PAMI*, 17(9):903–909, 1995.

[10] P. P. Roy, U. Pal, and J. Lladós. Recognition of multi-oriented touching characters in graphical documents. *ICVGIP*, 1:297–304, 2008.

[11] P. P. Roy, U. Pal, J. Lladós, and F. Kimura. Convex hull based approach for multi-oriented character recognition from graphical documents. *ICPR*, 2008.

[12] J. Song, Z. Li, M. R. Lyu, and S. Cai. Recognition of merged characters based on forepart prediction, necessity-sufficiency matching, and character-adaptive masking. *IEEE Trans. SMC B*, 35:2–11, 2005.

[13] Y.K.Chen and J.F.Wang. Segmentation of single- or multiple-touching handwritten numeral string using background and foreground analysis. *PAMI*, 22(11):1304–1317, 2000.