2011 International Conference on Document Analysis and Recognition

Template based Segmentation of Touching Components in Handwritten Text Lines

Le Kang and David Doermann
Institute for Advanced Computer Studies, University of Maryland
College Park, MD, USA
{*lekang, doermann*}@*umiacs.umd.edu*

***Abstract*—In this paper, we present a template based approach to the segmentation of touching components in handwritten text lines. Local patches around touching components are identified and a dictionary is created consisting of template patches together with their correct segmentations. We use two shape context based methods to compute similarity between input patches and dictionary templates to find the best match. The template's known segmentation is then transformed to segment the input patch. Experiments are carried on a dataset of touching text lines.**

***Keywords*-touching; segmentation; dictionary; shape context; thin-plate-spline; inner-distance**

I. Introduction

Segmentation at the line level is typically required by handwritten text recognition system, and when touching components affect line segmentation, recognition failure may result. The separation of touching components in general has been addressed extensively in the literature where they are typically segmented according to character size, profile, or junction points [1]. There are two primary approaches to the segmentation of touching characters, recognition-free and recognition-based. Recognition-free segmentation techniques use contour, skeleton and projection profile analysis[2-10] and use only structural information. These empirical methods may efficiently address some touching problems but may not be robust enough to handle a lot of variation or uncertainty. Recognition-based segmentation usually generates multiple candidate segmentation hypotheses and selects the optimal one based on recognition [11][12][13] or other evaluation functions [14][15][16]. These methods incorporate recognition-free segmentation and succeed only when the correct segmentation is in one of the candidates.

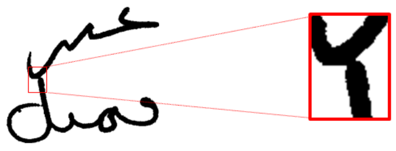


Figure 1: Local Touching Patch (in red box)

Despite a great deal of irregularity in touching handwritten text, we feel knowledge of the language can be exploited to define a trainable approach to segmentation. Some researchers categorize touching patterns [5][6][9] and make specific rules for handling each. In reality, more configurations are often possible, and it is necessary to generalize the categorization and corresponding strategy.

In this paper we propose a template based segmentation framework for touching handwritten text. In our approach we focus on the stage after text line detection by assuming the method of Kumar[17] is able to obtain Local Touching Patches (LTPs), as illustrated in Fig. 1. An LTP is the localized bitmap containing the touching strokes and usually only covers part of connected components shared by two different lines. Inspired by the localization of human body joints in [18], the proposed approach is based on the fact that if we know the correct segmentations of dictionary templates through training, we can segment similar LTPs accordingly. The key is that a common strategy must be applied, instead of designing individual rules for individual templates. In this way, segmentation can be trained for different languages, for example. The proposed approach uses a training and testing framework. In the training stage a dictionary is built from template LTPs and their correct segmentations. A correct segmentation is expressed as two patches of isolated components that comprise the LTP. In the testing stage, an LTP is compared to templates in the dictionary using shape context [19] or inner-distance shape context [20] to find the most similar one. Thin-plate-spline (TPS) [21] transform is calculated from the template to the input LTP to transform that template's known segmentation and obtain the input LTP's segmentation.

II. Proposed Approach

A. Dictionary Construction

The dictionary is built with the intent of covering the maximum possible number of touching configurations and is built to minimize the time for a input LTP to find the most similar template. Typically, this will be language dependent. In the proposed approach it is computationally expensive to obtain similarity between two LTPs. To reduce the computation load, we employ vector quantization [22]. Templates in dictionary are organized into clusters through Affinity Propagation [23], and clusters and their exemplars (i.e. centroids) are deterministically generated

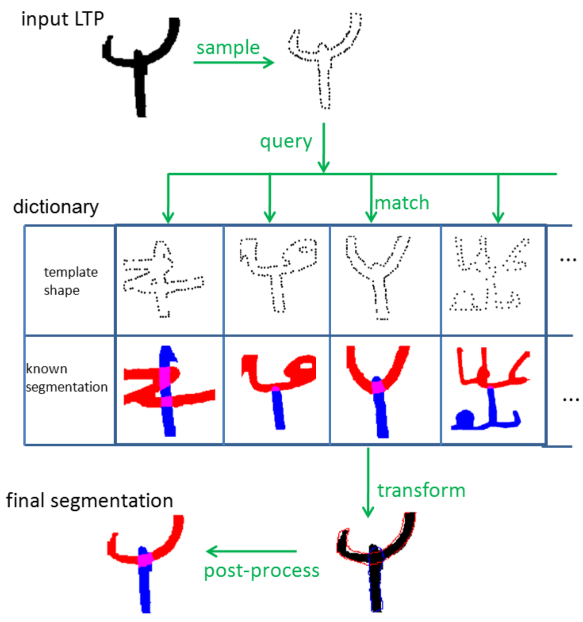


Figure 2: Process of template based segmentation

without prescribing the number of clusters. This is preferable because it is impossible to tell how many clusters should be obtained from the touching patterns. For an input LTP we first find the exemplar that has the greatest similarity, and then search the corresponding cluster to find the most similar template. This two-level structure reduces the search time from $O(N)$ to $O(N^{\frac{1}{2}})$. To further improve efficiency, a multi-level structure can be applied.

Exemplars naturally serve as representatives of different touching configurations. It is reasonable to do segmentation with a dictionary containing only exemplars, which is economical and reduces the chance of overfitting. In the experiments section we will compare the performance of the two kinds of dictionaries, full and exemplars-only.

B. *Shape Matching*

For matching two LTPs, we compare two methods: shape contexts with thin-plate-spline (SC+TPS) by Belongie et al.[19] and inner-distance shape context with dynamic programming (IDSC+DP) by Lin et al.[20].

Shape context characterizes the spatial distribution of sample points on the contour of a shape. For each point, it counts the number of other points in a uniform log-polar lattice and produces a histogram. Through distance normalization and relative orientation, it achieves scale and rotation invariance in describing shapes. The distance between two shape context histograms is defined by a χ^2 test. For shape comparison, an iterative scheme comprised of bipartite matching and TPS is used. The distance D_{SC+TPS} between two shapes is measured as a weighted sum of three terms

$$D_{SC+TPS} = 1.6D_{ac} + D_{sc} + 0.3D_{be} \tag{1}$$

where D_{ac} denotes appearance cost, D_{sc} denotes shape context cost, and D_{be} denotes TPS bending energy.

The inner-distance shape context augments the shape context for better description of part structure and articulation. It replaces the Euclidean distance with the length of shortest path within the shape boundary. Dynamic programming is used to match two sets of points and the distance $D_{IDSC+DP}$ between two shapes is determined only by the inner-distance shape context cost D_{idsc}

$$D_{IDSC+DP} = D_{idsc} \tag{2}$$

In the proposed segmentation framework, shape similarity S is defined as the negation of the shape distance

$$S = -D_{SC+TPS} \quad or \quad S = -D_{IDSC+DP} \tag{3}$$

which is intuitive and convenient for the affinity propagation clustering.

C. *Segmentation*

During the segmentation stage, we hierarchically compare the dictionary templates and the input LTPs. Input LTPs are segmented according to the segmentation of its matching templates. Assume P_{input} and $P_{template}$ are the point sets of input LTP and its best matching template respectively. Through a few iterations of bipartite matching and TPS we find the parameter θ of TPS transform $T(\cdot)$ that warps $P_{template}$ into P_{input}

$$P_{input} \approx T(\theta;\ P_{template}) \tag{4}$$

where $\approx$ means resemblance with tolerance to some error. Let C_a and C_b denote contours of two isolated components in template's segmentation. Then the deformed contours C'_a and C'_b are defined as

$$C'_a = T(\theta;\ C_a) \tag{5}$$
$$C'_b = T(\theta;\ C_b) \tag{6}$$

Then a natural judgment is that the foreground pixels of input LTP that fall into different deformed contours belong to different components, and the overlapping region belongs to both components. By allowing two components to share a common area, we have a better chance of restoring the original appearance of the text, compared to cutting the foreground into non-overlapping parts. We assign those pixels outside both deformed contours to the nearest stroke. Fig. 2 illustrates the segmentation process.

The advantage of this segmentation strategy is that it handles all cases in one framework, rather than applying different rules for different configurations. In addition, scaling to new patterns in the dictionary is easy in this framework. When new touching types are observed, we simply add sample LTPs to the dictionary and rerun clustering .

III. Experimental results

Previously there were no available handwritten text dataset that has ground truth for touching or overlapping components, so we created a dataset by varying the distance between existing text lines and recorded interactions. The source is 250 Arabic handwritten text binary images with ground truth. We reduced the spacing between text lines in each image until neighboring lines touched each other, in the similar fashion employed by Kumar et al.[24], and extracted LTPs with size 120×120 at where touching occurred. The patch size is fixed for simplicity, since the stoke width is quite consistent across the dataset. Each correct segmentation includes two isolated component patches which have same size as the LTPs, created at the same time according to the pixel level ground truth for the source images.

From these LTPs we form a training set and a testing set, each having 744 samples. We set parameters for SC+TPS and IDSC+DP according to [19] and [20]. But unlike [19], our experiments use a distance between shapes that only involves the shape context cost and bending energy

$$D_{SC+TPS} = D_{sc} + 0.3 D_{be} \tag{7}$$

The appearance cost is not used since we believe these binary document images contain very little color and texture information, and computing appearance cost brings relatively large work load.

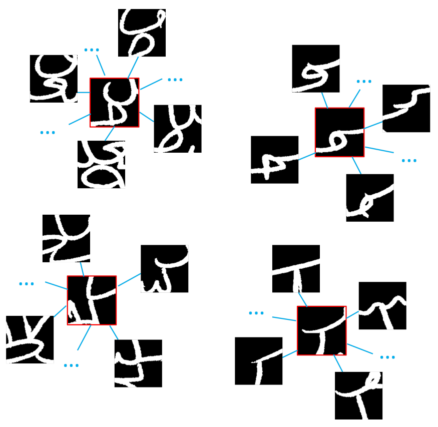
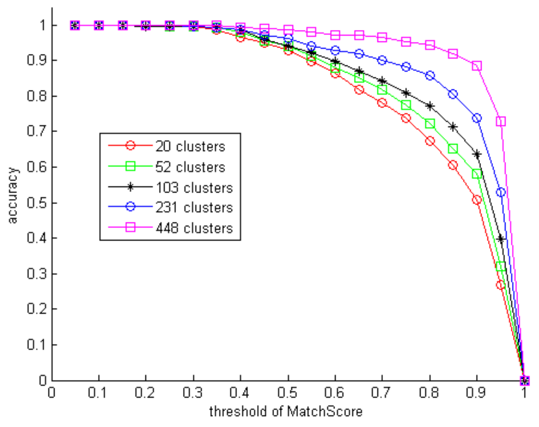
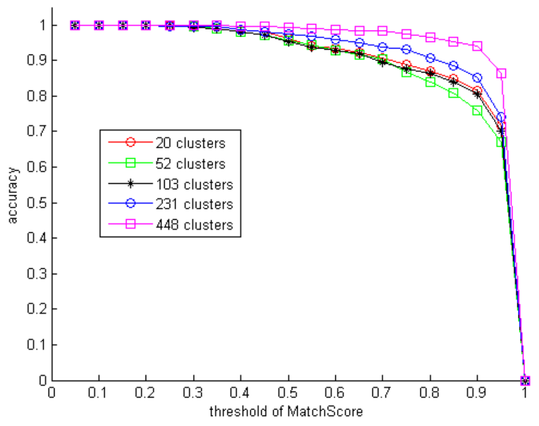


Figure 3: Template LTPs are organized into clusters.

Evaluation of segmentation performance is not straightforward. Even if a few pixels are wrongly assigned, it may still be a good segmentation. An indirect but reasonable rule is that the segmentation is acceptable if recognition gives the correct result, but this requires a very accurate recognizer. In practice we use the MatchScore method [25]. Assume the input LTP is segmented into two components Q_A and Q_B, each in an image of the same size as the input LTP. They are compared to the input LTP's ground truth segmentation which are components of G_A and G_B respectively. For each



(a)



(b)

Figure 4: Accuracy under 5 numbers of clusters on (a)exemplar-only dictionary (b) full dictionary.

LTP's segmentation result we compute the MatchScore (MS) that is defined as :

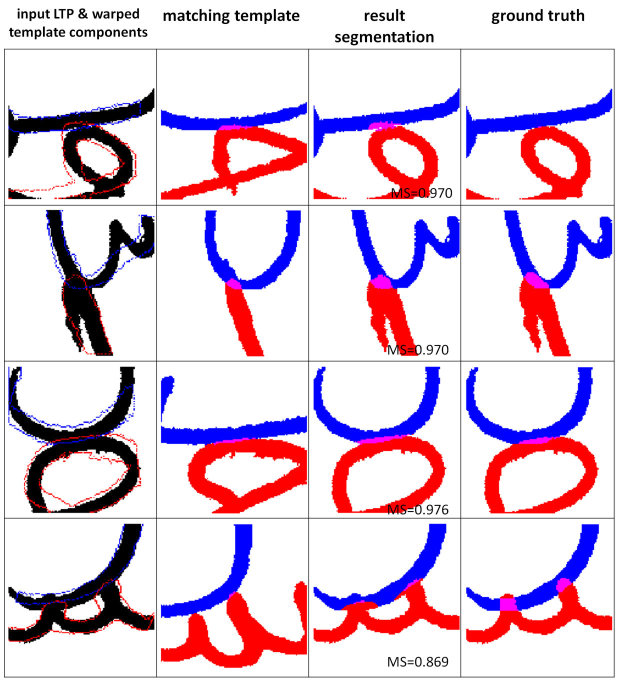
$$MatchScore = \frac{2 \times MS_A \times MS_B}{MS_A + MS_B} \tag{8}$$

$$MS_A = \frac{area(Q_A \bigcap G_A)}{area(Q_A \bigcup G_A)} \tag{9}$$

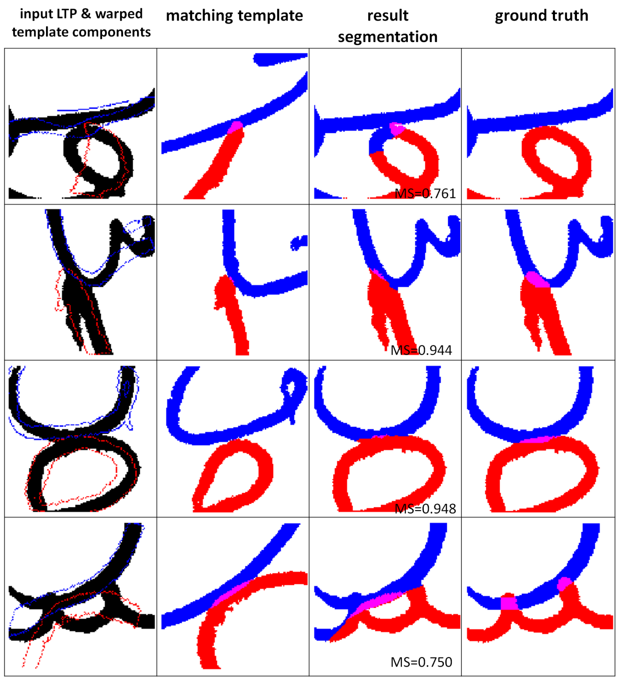
$$MS_B = \frac{area(Q_B \bigcap G_B)}{area(Q_B \bigcup G_B)} \tag{10}$$

where $area(\cdot)$ calculates the foreground area of a binary image patch. We set a threshold for the MatchScore, where values greater than the threshold are treated as correct segmentation and less than it incorrect. Define accuracy as the ratio of correctly segmented LTPs among all testing LTPs, and the accuracy is expected to rise as the threshold of MatchScore decreases.

We explore the segmentation performance on two kinds of dictionaries with different number of clusters, and then



(a)



(b)

Figure 5: (a)Correct($MatchScore > 0.80$) segmentation samples by SC+TPS. (b)Corresponding results by IDSC+DP. Purple areas are shared by both blue and red components.

analyze the results.

1) Number of Clusters: For simplicity, only SC+TPS is used to obtain similarities between training shapes and to construct the dictionary. Through clustering, similar template LTPs are grouped together, as shown in Fig. 4. Quantizing the dictionary improves search efficiency but may lead to non-globally optimal matching, thus it is worth careful consideration. The number of clusters can be adjusted in Affinity Propagation by modifying the preference values for the input samples. The total number of training samples is fixed, thus a greater number of clusters corresponds to a smaller average cluster size. We ran the SC+TPS shape matching and segmentation on the training data itself under five different numbers of clusters for both full and exemplar-only dictionary. Accuracy curves are shown in Fig. 4. We can see for exemplar-only dictionaries the accuracy is higher when more clusters are used. However in full dictionaries the worst performance occurs with 52 clusters rather than 20. This is reasonable since as the number of clusters decreases, more in-cluster searches will be conducted. The two extremes are equivalent: a single cluster containing all samples and each sample being a cluster. In the following experiments we choose the 231 cluster dictionary, which is a trade-off between performance and efficiency.

2) Full Dictionary vs. Exemplar-only: Table I shows the results on the testing set. Although the full dictionary performs better than the exemplar-only dictionary on training set according to Fig. 4, no significant advantage is observed from the results of testing set, and for SC+TPS the exemplar-only beats the full dictionary. We suspect this is because the full dictionary overfits the training data. The results also demonstrate that exemplars are capable of characterizing touching configurations and therefore using a full dictionary is unnecessary. We also notice that SC+TPS outperforms IDSC+DP at this task.

Table I: Test Results with MatchScore Threshold 0.80

dictionary	accuracy	
	SC+TPS	IDSC+DP
full	0.696	0.563
exemplar-only	0.714	0.555

3) Correct Segmentations and Errors: Fig. 5 shows several correct ($MatchScore \geq 0.80$) segmentation samples obtained through SC+TPS and their corresponding results with IDSC+DP. From Fig. 5(a) we see the templates' segmentations are properly transformed to mark conterparts in input LTPs. Despite disparities between input LTPs and matching templates, reasonable segmentations towards different touching types are obtained. Comparing Fig. 5(a) and (b), we see SC+TPS indeed finds better templates than IDSC+DP. There are also a number of segmentation errors($MatchScore < 0.80$), as illustrated in Fig. 6. The reasons are fourfold. First, for a very unusual input LTP no proper template can be found. Second, ambiguity exists so

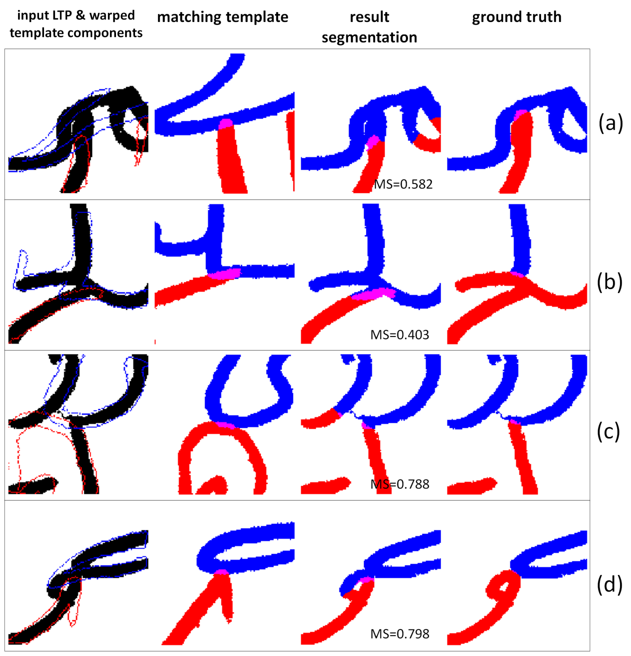


Figure 6: Four kinds of segmentation errors. (a) Bad template. (b)Ambiguity. (c)Noise (disturbing components). (d) Transform deviation.

that the most similar template may not be the right one. Third, noise (disturbing components) causes mismatches. Fourth, although a similar template is found, the TPS transform can not mark corresponding components very accurately.

IV. Conclusion

We have presented a new approach for segmenting touching components in handwritten text lines. The key point of our approach is to build a dictionary of local touching patches and transform the template's segmentation to segment input touching patch. Experiments show various touching patches are reasonably segmented, and cluster exemplars are enough for constructing a dictionary that represent different touching types.

Acknowledgment

The partial support of this research by DARPA through BBN/DARPA Award HR0011-08-C-0004 under subcontract 9500009235, the US Government through NSF Award IIS-0812111 is gratefully acknowledged.

References

[1] L. Likforman-Sulem, A. Zahour and B. Taconet. Text line segmentation of historical documents: a survey. *IJDAR*, vol. 9, no. 2, pp. 123-138, 2007.
[2] R. G. Casey and E. Lecolinet. A survey of methods and strategies in character segmentation. *TPAMI*, vol. 18, no. 7, pp. 690-706, 1996.
[3] V. Papavassiliou, T. Stafylakis, V. Katsouros, and G. Carayannis. Handwritten document image segmentation into text lines and words. *PR*, 43(1):369-377, 2010.
[4] N. Tripathy, and U. Pal. Handwriting segmentation of unconstrained Oriya text. *9th IWFHR*, pp. 306-311, 2004.
[5] H. Ikeda, Y. Ogawa, M. Koga, H. Nishimura, H. Sako, and H. Fujisawa. A recognition method for touching Japanese handwritten characters. *ICDAR'09*, pp.641-644, 1999.
[6] K.K. Kim, J.H. Kim, and C.Y. Suen. Recognition of unconstrained handwritten numeral strings by composite segmentation method. *ICPR'00*, 2:594-597, 2000.
[7] S. Wshah, and Z. Shi, V. Govindaraju. Segmentation of Arabic handwriting based on both contour and skeleton segmentation. *ICDAR'09*, pp.793-797, 2009.
[8] Z. Shi, and V. Govindaraju. Segmentation and recognition of connected handwritten numeral strings. *PR*, 30(9):1501-1504, 1997.
[9] Y.K. Chen, and J.F. Wang. Segmentation of single- or multiple-touching handwritten numeral string using background and foreground analysis. *TPAMI*, 22(11):1304-1317, 2000.
[10] T. Yamaguchi, T. Yoshikawa, T. Shinogi, S. Tsuruoka, and M. Teramoto. A segmentation method for touching Japanese handwritten characters based on connecting condition of lines. *ICDAR'01*, pp.837-841, 2001.
[11] Y. Lei, C.S. Liu, X.Q. Ding, Q. Fu. A recognition based system for segmentation of touching handwritten numeral strings. *IWFHR'04*, pp. 294-299, 2004.
[12] C.L. Liu, H. Sako, H. Fujisawa. Effects of classifier structures and training regimes on integrated segmentation and recognition of handwritten numeral strings. *TPAMI*, 26(11):1395-1407, 2004.
[13] Y.H. Tseng and H.J. Lee. Recognition-based handwritten Chinese character segmentation using a probabilistic viterbi algorithm. *Pattern Recog. Let.*, 20(8):791-806, 1999.
[14] E. Vellasques, L.S. Oliveira, R. Sabourin, A.S. Britto, and A.L. Koerich. Modeling segmentation cuts using support vector machines. *IWFHR'06*, pp. 41-46, 2006.
[15] D. You and G. Kim. An approach for locating segmentation points of handwritten digit strings using a neural network. *ICDAR'03*, 1:142-146, 2003.
[16] Y.J. Wang, X.B. Liu and Y.D. Jia. Statistical modeling and learning for recognition-based handwritten numeral string segmentation. *ICDAR'09*, pp. 421-425, 2009.
[17] J. Kumar, L. Kang and D. Doermann. Segmentation of handwritten text lines in presence of touching components. *ICDAR'11*, 2011, in press.
[18] G. Mori, J. Malik. Recovering 3D human body configurations using shape contexts. *TPAMI*, 28(7):1052-1062, 2006.
[19] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *TPAMI*, 24(4):509-522, 2002.
[20] H. Ling, D.W. Jacobs. Shape classification using the inner-distance. *TPAMI*, 29(2):286-299, 2007.
[21] F.L. Bookstein. Principal warps: Thin-Plate-Splines and decomposition of deformations. *TPAMI*, 11(6):567-585, 1989.
[22] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic, 1991.
[23] B.J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315:972-976, 2007.
[24] J. Kumar, W. Abd-Almageed, L. Kang, and D. Doermann. Handwritten Arabic text line segmentation using affinity propagation. *DAS'10*, pp. 135-142, 2010.
[25] B. Gatos, N. Stamatopoulos, G. Louloudis. ICDAR 2009 handwriting segmentation contest. *ICDAR'09*, 2009