2014 14th International Conference on Frontiers in Handwriting Recognition

Evaluation of Geometric Context Models for Handwritten Numeral String Recognition

Yi-Chao Wu, Fei Yin, Cheng-Lin Liu
National Laboratory of Pattern Recognition (NLPR)
Institute of Automation, Chinese Academy of Sciences
Beijing 100190, China
{yichao.wu, fyin, liucl}@nlpr.ia.ac.cn

***Abstract*—Character string recognition based on over-segmentation by integrating character classifier and context models has been demonstrated successful. Geometric context models characterizing the candidate character likeliness and between-character relationship have shown benefits in several scripts but have not been evaluated in numeral string recognition. Compared with Chinese scripts mixed with alphanumeric and punctuation marks, numeral strings are less variant in character outline and between-character relationship. This study, via evaluating geometric context models used in Chinese handwritten text recognition, shows that geometric context is beneficial to handwritten numeral string recognition as well. Particularly, we propose an improved binary geometric model that combines single-character and between-character features such that the model functions like a bi-character classifier. Combining this binary geometric model with unary geometric model and character classifier, we obtain significant improvement of numeral string recognition performance on the NIST SD-19.**

***Keywords*—numeral string recognition; integrated segmentation and recognition; geometric context models**

I. Introduction

Handwritten numeral string recognition has been intensively studied due to its wide applications such as bank check reading, form processing, and zip code recognition. However, it remains a challenging problem due to the diversity of writing styles, the character segmentation difficulty and the strict requirement of recognition performance. For the last 30 years, many efforts have been done with the aim to build a successful numeral string recognition system. Ha et al. built an off-line handwritten numeral string recognition system by combining segmentation-based and segmentation-free methods [1]. By using background and foreground analysis, Chen et al. significantly improved the correct rate of segmenting single- or multiple-touching handwritten numeral string [2]. Moreover, a recognition and verification strategy [3], decision value generator [4] and geometric contextual knowledge [5] were also used to improve the recognition performance, respectively.

Our numeral string recognition system is based on over-segmentation by integrating character classifier and geometric context models, and this framework has been demonstrated successful in handwritten string recognition [6], [7] and transcript mapping [8]. In an integrated segmentation-recognition system, the string image is first over-segmented into primitive segments, consecutive segments are concatenated into candidate character patterns, which are assigned candidate classes and confidence scores by a character classifier. Combining the class scores with context scores, the final segmentation is obtained by searching for the optimal path with the maximum score or minimum cost. Since the candidate patterns formed by concatenating primitive segments include both character and non-character patterns, the classifier and context models should be not only precise to classify characters but also resistant to non-characters. Therefore, a lot of efforts have been devoted to solve this problem from two perspectives as below.

Many researchers focused on designing a character classifier which is resistant to non-characters. Some further studies show that it is crucial for discriminative models, such as neural classifiers [9], [10], [11] and support vector classifiers [7], etc. to be trained with non-character samples. However, non-character training shows limited influence on the performance of generative models that are assumed to be inherently resistant to non-characters [7].

Other researchers integrated geometric context with the character classifier to evaluate the segmentation paths, where the geometric context helps disambiguate the character segmentation. Although only some outline features, such as character size and projection properties, etc. were used in [12], [13], [14], they significantly improved the recognition performance. In order to describe the geometric context more adequately, some researchers further elaborated geometric context into unary and binary, character class-dependent and class-independent geometric models. The elaborated geometric context models have been successfully used in character string recognition [6], [15], [16], and transcript mapping [8]. However, geometric context for handwritten numeral string has not been investigated adequately.

In this study, we focus on the design of geometric context models for numeral string recognition. We evaluate the geometric models originating from Chinese text recognition presented in [8]. Further, we propose an improved binary geometric model that combines single-character and between-character features such that the model functions like a bi-character classifier. Combining this binary geometric model with unary geometric model and character classifier, we obtained significant improvement of numeral string recognition performance on the NIST Special Database 19 (SD-19).

The rest of this paper is organized as follows: Section II gives an overview of the numeral string recognition system. Section III describes the geometric context models. Section IV presents the experimental results on numeral string recognition. Finally, the paper ends with concluding remarks in Section V.

2167-6445/14 $31.00 © 2014 IEEE
DOI 10.1109/ICFHR.2014.40

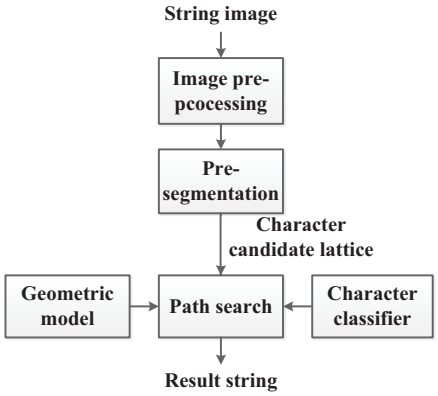

Fig. 1. Diagram of numeral string recognition system.

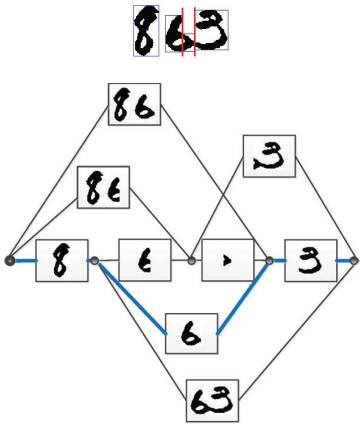

Fig. 2. Segmentation candidate lattice of a numeral string image. The path corresponding to the most plausible segmentation is denoted by a thick line.

II. Numeral String Recognition System

The diagram of our numeral string recognition system is shown in Fig. 1. In image pre-processing, underlines are detected from string images and removed. In pre-segmentation (over-segmentation), the connected components are firstly slant-corrected, then those heavily overlap in horizontal direction are merged, finally the components with large width (relative to the estimated string height) or abnormally large width/height ratio are regarded as potential touching patterns and are split by upper/lower profile curve analysis to generate vertical cuts [7]. After that, the string image is represented as a sequence of primitive image segments. Consecutive primitive segments are combined to generate candidate character patterns, forming a segmentation candidate lattice as shown in Fig. 2. Each path from the start node to the end node corresponds to a candidate segmentation. Thus, the rest of the task is to find the optimal path with minimum cost or maximum score.

In the system, the input string sample is over-segmented and decomposed to be sequences of candidate characters, each denoted by $X = x_1...x_m$. Each candidate character is assigned candidate class (denoted as c_i) by a character classifier, and then the result of character string recognition is a character string $C = c_1...c_m$. We adopt the path evaluation criterion presented in [6] which formulates string recognition from the view of Bayesian decision. For saving space, we give the criterion directly and more details can be found in [6].

Denote the score of classifying character x into class c given by the character classifier as $P(c|x)$. The unary class-dependent geometric (**ucg**) score, unary class-independent geometric (**uig**) score, binary class-dependent geometric (**bcg**) score and binary class-independent geometric (**big**) score are denoted as $P(c_i|g^{ucg})$, $P(z_i^p = 1|g^{uig})$, $P(c_{i-1}, c_i|g^{bcg})$, and $P(z_i^g = 1|g^{big})$, respectively, where g denotes corresponding geometric feature and the output scores are given by geometric models classifying on features extracted. Among them, $P(z_i^p = 1|g^{uig})$ indicates the probability of the candidate character being a valid character and $P(z_i^g = 1|g^{big})$ indicates the probability of the gap between two successive candidate characters being a between-character gap. All the four geometric context models will be elaborated in Section III. We obtain a logarithm likelihood function $f(X, C)$ for the segmentation-recognition path evaluation:

$$f(X,C) = \sum_{i=1}^{m}(w_i \cdot \log P(c_i|x_i) + \lambda_1 \cdot \log P(c_i|g_i^{ucg}) + \lambda_2 \cdot \log P(z_i^p = 1|g_i^{uig}) + \lambda_3 \cdot \log P(c_{i-1}, c_i|g_i^{bcg}) + \lambda_4 \cdot \log P(z_i^g = 1|g_i^{big})), \quad (1)$$

where w_i is the word insertion penalty which is used to overcome the bias to short strings, while we use Weighting with primitive Segments Number (WSN) [6] in the system, $\lambda_1 - \lambda_4$ are the weights to balance the effects of different models optimized with Maximum Character Accuracy (MCA) criterion [6]. As can be seen, compared with [6], the language model is dropped off since little linguistic context is available in numeral strings. Via confidence transformation, the five models consisting of one character classifier and four geometric models are combined to evaluate the segmentation paths. We use the sigmoidal confidence transformation to convert the classifier outputs to posterior probabilities. The confidence parameters are estimated by minimizing the cross entropy (CE) loss function, which is commonly used in logistic regression and neural network training, on a validation dataset (preferably different from the dataset for training classifiers) [17]. As for the path search, a beam search algorithm [18] is used. The search proceeds in frame-synchronous fashion and in order to accelerate search, we only retain a limited number of partial paths with maximum scores at each frame.

III. Geometric Context Models

Geometric context models have not been evaluated in numeral string recognition, where they should play a more important role to exclude non-characters and further improve the recognition accuracy. In this study, we adopt the framework of geometric context model presented in [8], where geometric context is divided into four statistical models (unary and binary class-dependent, unary and binary class-independent), abbreviated as "**ucg**", "**bcg**", "**uig**", "**big**", respectively. The details are in the following.

A. Class-dependent Geometric Features

The class-dependent geometric model can be seen as a complement to the character classifier since the candidate patterns retain their original outlines without normalization designed for single-character classifier, which may delete some useful context information related to writing styles.

For modeling **ucg**, we use the same 42 geometric features from a candidate character pattern as those used in [8], which are grouped into three categories: 10 scalar features related to the bounding box of the character, four scalar features related to the vertical center of text line, and 28 profiles-based features inspired by the methods of [19], [20].

For **bcg**, 24 features are firstly extracted [8], which can be grouped into two categories: 16 scalar features between the bounding boxes of two consecutive character patterns, eight features between the profiles of two consecutive character patterns.

In addition, we combine the 42 single-character geometric features of two consecutive patterns with the original 24 between-character features, to form a 108-D feature vector. The model using the 108 features functions like a bi-character classifier which classifies the pair of neighboring characters. We will show in experiments that this **bcg** model combining single-character and between-character features performs much better than the original binary geometric model.

B. Class-independent Geometric Features

The unary and binary class-independent geometric models use the same features as those in [8], which consist of 12 and 14 features, respectively. The **uig** model is used to measure whether a candidate pattern is a valid character or not, while the **big** model is used to measure whether an over-segmentation gap is a valid between-character gap or not.

C. Statistical Models

For modeling the class-dependent geometry, we firstly cluster the character classes into $M(M < 10)$ super-classes using the K-means clustering algorithm so as to reduce the number of character geometry classes while maintaining the classification performance. The optimal number of M will be discussed in Section IV. After clustering, each single character is assigned to one of M super-classes and a pair of successive characters thus belongs to one of $M \times M$ binary super-classes. We use a quadratic discriminant function (QDF) for classification for both **ucg** and **bcg** models. For the convenience of comparison, the **ucg** feature remains 42-dimension with M-class QDF, while for **bcg**, the performances in both original space and subspace projected by Fisher linear discriminant analysis (FLDA) or principal components analysis (PCA) are evaluated in Section IV.

The modeling of class-independent geometry is actually a two-class problem. We use a linear support machine (SVM) trained with character and non-character samples for **uig** and one trained with valid and invalid gaps for **big**.

IV. Experimental Results

We evaluated the performance of numeral string recognition on the NIST Special Database 19 (SD-19). The character classifier was trained with isolated digit samples and then applied to numeral string recognition.

A. Isolated Digit Recognition

As in previous evaluation studies [7], [21], the training set for character classifier comprises of the 66,214 digit patterns of 600 writers (no. 0-399 and no. 2100-2299), and the test set comprises of the 45,398 digit patterns of 400 writers (no. 500-699 and no. 2400-2599). Before feeding to the classifier, the character image was first normalized using moment normalization [22], then gradient direction histogram feature was extracted using the method of normalization-cooperated gradient feature (NCGF) [23]. The obtained 200D feature vector was reduced to 100D by principal component analysis (PCA), and then input into the the MQDF (Modified Quadratic Discriminant Function) [24] classifier. We obtained the accuracy of 98.80% on our test set. As this work is primarily to evaluate the effects of geometric models, we did not try to optimize the accuracy of the character classifier, and did not train the classifier with non-characters.

For confidence transformation parameter estimation, we first trained a classifier on $4/5$ of training samples, and the classifier outputs on the heldout $1/5$ of training samples were used to estimate the confidence parameters.

B. Numeral String Recognition

To compare with previous results, we evaluated string recognition performances on the same set of numeral strings of writers no.1800-2099 in the NIST SD-19 as the ones used in [7], [21], which contains 1476 3-digit strings and 1471 6-digit strings. Instead of reporting recognition rates for two string lengths separately, we report them on the whole test set of 2,947 strings. For comparison, the two accuracies in [7] can be combined into one manually.

For training the geometric context models, we used the same training set as that used in character classifier training. After discarding pages with faded ink or low quality, we obtained 14,831 strings from the pages of 600 writers, and used the candidate characters generated by over-segmenting these strings or neighboring character pairs for training the geometric models. Similarly, 4/5 of randomly chosen training samples were used to train the classifiers of geometric models, and the rest were used to estimate the confidence parameters. Finally, all of the pages were used to train the fusion weights of different models in (1) with MCA. It should be mentioned that the training samples for geometric models were all slant-corrected to make them compatible with test samples, but the isolated digit training samples were not slant-corrected. However, even with such an "incompatible" classifier, we obtained encouraging results as shown in this section.

In addition to string-level error rate, we report the recognition performance using two character-level accuracy metrics following [25]: Correct Rate (CR) and Accurate Rate (AR):

$$\begin{aligned} CR &= (N_t - D_e - S_s)/N_t, \\ AR &= (N_t - D_e - S_s - I_e)/N_t, \end{aligned} \tag{2}$$

where N_t is the total number of characters in the transcript of test strings. The number of substitution errors (S_e), deletion errors (D_e) and insertion errors (I_e) are calculated by aligning the recognition result string with the transcript by dynamic programming.

In the following, we will show how geometric context improves the performance. For the convenience of comparison, we will separately discuss the performance of class-independent models (**uig**, **bcg**) and class-dependent models (**ucg**, **bcg**). It should be noted that the character classifier (MQDF) and the union of all geometric models are abbreviated as **cls** and **g**, respectively.

1) Class-independent Geometric Model: As stated in Section III, **uig** indicates whether a candidate pattern is a valid character or not and **big** indicates whether a segmentation point between two adjacent primitive segments is a between-character gap or not. The effects of different combinations are shown in Table I, where cls denotes the character classifier. It is observed that **big** yields larger improvement than **uig**, which does little to increase the accuracy. This is because **uig** models the outline shapes of digits, which do not differ significantly from those of non-characters.

TABLE I. EFFECTS OF CLASS-INDEPENDENT GEOMETRIC MODELS.

Combination	CR (%)	AR (%)	Error (%)
cls	99.25	98.88	3.66
cls+uig	99.25	98.88	3.66
cls+big	99.28	99.07	3.19
cls+uig+big	99.28	99.07	3.19

2) Class-dependent Geometric Model: The performance of class-dependent geometric context is highly related to the choice of super-classes. To decide the partition of super-classes, we extracted 42D **ucg** feature on the training set and used the k-means algorithm to to cluster the digit classes into several super-classes ranging from three to seven. After clustering, we manually adjusted some super-classes to make the partition more consistent. The partitions are listed in Table II. It can be observed that the partitions of 4-7 super-classes are consistent in style, whereas it is too difficult to divide the digits into 3 super-classes consistently.

We first investigated the effects of feature dimensionality reduction on class-dependent geometric models. For the original **bcg** and **ucg** features are 24D and 42D (concatenation of two character features results in 84D), respectively, we reduced our improved "bi-character" 108D **bcg** feature to 24D, 84D by PCA and to $(C-1)$D by FLDA, where C denotes the number of classes. We also evaluated the performance of 84D single-character **bcg** feature, i.e. combining 42D **ucg** features of the two consecutive patterns. For saving space, we only list the highest string-level accuracy among all the combinations of class-dependent models (**cls+ucg**, **cls+bcg**, **cls+ucg+bcg**, **cls+g**), as shown in Table III. It can be seen that the 84D single-character **bcg** performs only a little worse than 108D feature and outperforms the 24D **bcg**. This justifies the effectiveness of the "bi-character" feature. As for feature reduction, the 84D **bcg** by PCA is inferior to 84D single-character **bcg**, and the 24D **bcg** by PCA yields better results than the original 24D **bcg** but is worse than both of the two 84D **bcg** models. The $(C-1)$D **bcg** by FLDA yields the results only slightly worse than the 108D **bcg**. It shows that all the models related to the bi-character classifier is better than the original one in [8], and the 108D **bcg** without dimensionality reduction performs best among them due to its combination with single-character and between-character features.

TABLE II. PARTITIONS OF SUPER-CLASSES.

#Super-class	Pattitions
3	**0 1 8**
	2 3 5 6
	4 7 9
4	**0 3 5 8**
	1
	2 6
	4 7 9
5	**0 3 8**
	1
	2 6
	4 7 9
	5
6	**0 3 8**
	1
	2 6
	4
	5
	7 9
7	**0 8**
	1
	2 6
	3
	4
	5
	7 9

TABLE III. LOWEST STRING-LEVEL ERROR RATES (%) WHEN COMBINING GEOMETRIC CONTEXT WITH DIFFERENT **BCG** MODELS.

#Super-class	24D bcg	108D bcg	84D bcg	24D bcg (PCA)	84D bcg (PCA)	(C-1)D bcg (LDA)
3	3.19	3.12	3.16	3.19	3.19	3.09
4	3.05	2.82	2.85	2.95	2.85	2.88
5	2.99	2.71	2.85	2.99	2.92	2.99
6	3.09	2.68	2.68	2.82	2.78	2.78
7	2.88	2.68	2.71	2.82	2.75	2.78

To further investigate the class-dependent models, we compared the best 108D **bcg** model with the original one in [8] and evaluated all the five partitions of super-classes. The results are shown in Table IV and Table V. From both of them, we find that sometimes a single geometric model performs better than multiple geometric models, but mostly, the combination of all geometric models gives the best result.

The results in two tables show that the choice of super-class is important to the final result. In both cases, the partition of three super-classes performs worst, while the best result is given by the partition of six or seven super-classes. Further increasing the number of super-classes is not beneficial because on one hand, the performance of using six or seven super-classes is saturated, and on the other hand, a larger number of super-classes needs more training samples to overcome over-fitting. Comparing the results in Table IV and Table V, it can be seen that the performance of **bcg** has been greatly improved using our improved "bi-character" 108D feature. The results in Table IV show that the original **bcg** leads to little improvement of performance, whereas the proposed improved **bcg** performs

TABLE IV. EFFECTS OF DIFFERENT COMBINATIONS OF CLASS-DEPENDENT MODELS WITH **ORIGINAL BCG** FEATURE IN DIFFERENT SUPER-CLASSES.

#Super-class	Combination	AR (%)	CR (%)	Error (%)
none	cls	99.25	98.88	3.66
3	cls+ucg	99.28	99.03	3.33
	cls+bcg	99.25	98.88	3.66
	cls+ucg+bcg	99.28	99.03	3.33
	cls+g	99.28	99.07	3.19
4	cls+ucg	99.28	99.11	3.12
	cls+bcg	99.25	98.90	3.63
	cls+ucg+bcg	99.28	99.11	3.12
	cls+g	99.29	99.14	3.05
5	cls+ucg	99.27	99.13	3.05
	cls+bcg	99.25	98.89	3.63
	cls+ucg+bcg	99.28	99.12	3.09
	cls+g	99.29	99.15	2.99
6	cls+ucg	99.26	99.12	3.09
	cls+bcg	99.25	98.89	3.63
	cls+ucg+bcg	99.26	99.12	3.09
	cls+g	99.26	99.12	3.09
7	cls+ucg	99.28	99.15	2.95
	cls+bcg	99.25	98.89	3.63
	cls+ucg+bcg	99.28	99.15	2.99
	cls+g	99.28	99.17	2.88

TABLE V. EFFECTS OF DIFFERENT COMBINATIONS OF CLASS-DEPENDENT MODELS WITH **IMPROVED BCG** FEATURE IN DIFFERENT SUPER-CLASSES.

#Super-class	Combination	AR (%)	CR (%)	Error (%)
none	cls	99.25	98.88	3.66
3	cls+ucg	99.28	99.03	3.33
	cls+bcg	99.28	99.09	3.16
	cls+ucg+bcg	99.25	99.10	3.12
	cls+g	99.29	99.08	3.19
4	cls+ucg	99.28	99.11	3.12
	cls+bcg	99.30	99.22	2.82
	cls+ucg+bcg	99.29	99.19	2.85
	cls+g	99.29	99.19	2.85
5	cls+ucg	99.27	99.13	3.05
	cls+bcg	99.31	99.24	2.75
	cls+ucg+bcg	99.32	99.24	2.71
	cls+g	99.32	99.24	2.71
6	cls+ucg	99.26	99.12	3.09
	cls+bcg	99.34	99.26	2.68
	cls+ucg+bcg	99.32	99.22	2.75
	cls+g	99.32	99.22	2.75
7	cls+ucg	99.28	99.15	2.95
	cls+bcg	99.33	99.25	2.68
	cls+ucg+bcg	99.33	99.25	2.68
	cls+g	99.30	99.24	2.82

much better as shown in Table V. The proposed **bcg** combines single-character and between-character features such that the model functions like a bi-character classifier, which can be seen as a supplement to the single-character classifier. We compare the results in Tables I, IV, V, and find that generally the class-dependent models perform much better than the class-independent models except in the case of three super-classes.

The best results in our system and the others in [7], [21] are listed in Table VI, where SVC-rbf represents the support vector classifier with Gaussian kernel, DLQDF denotes the Discriminative Learning Quadratic Discriminant Function classifier and PC the Polynomial Classifier. It should be mentioned that all the other three classifiers are trained with both characters and non-characters and have test accuracies higher than 99.0%. Compared with the string error rate 3.66% given by MQDF classifier only, the error rate of our method combining geometric models is reduced by 26.78% relatively. The error rate we obtained is lower than that reported in [7] with SVC-rbf and is only 0.03% higher than the result of DLQDF (in fact that equals only one wrong string). There is still a gap in accuracy between our system and the one with PC, which has much higher classification accuracy than the MQDF. As a whole, the geometric models, especially the improved "bi-character" **bcg**, leads to significant improvement in our numeral string recognition system.

TABLE VI. THE COMPARISON OF BEST RESULTS OF DIFFERENT SYSTEMS.

Classifier (%)	CR (%)	AR (%)	Error (%)
cls	99.25	98.88	3.66
Ours	99.34	99.26	2.68
SVC-rbf [7]	–	–	3.22
DLQDF [21]	–	–	2.65
PC [21]	–	–	2.04

Fig. 3 shows some string samples which are recognized successfully after combining with geometric context. We can see that by combination with geometric context, not only the segmentation paths but also the digit classes are corrected. Fig. 4 shows some mis-recognition samples. The error mainly lies in three aspects: the first is the over-segmentation, some cuts are missing and others are not precisely located since we only generate vertical cuts; the second lies in the character classifier when the digits are not written in a normal style; the third is because of the low quality of some page images.

V. CONCLUSION

In this paper, we evaluated the effects of geometric context models in numeral string recognition. Experimental results show that class-dependent geometric models perform better than class-independent geometric models in numeral strings when choosing the super-classes properly. Particularly, we proposed an improved binary geometric model that combines single-character and between-character features such that the model functions like a bi-character classifier. Combining this binary geometric model with unary geometric model and character classifier, we obtained significant improvement of numeral string recognition performance. It is our future work to improve the over-segmentation of numerals and design a better classifier to combine with the geometric context models.

ACKNOWLEDGMENT

This work has been supported by the National Natural Science Foundation of China (NSFC) grants 61305005 and 61175221.

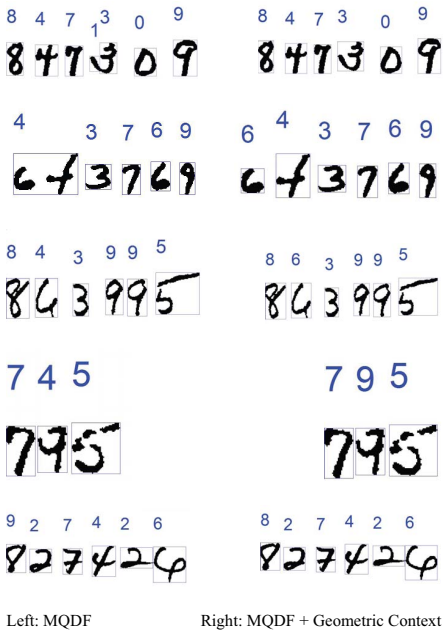

Fig. 3. Examples of how geometric context improve the recognition accuracy.

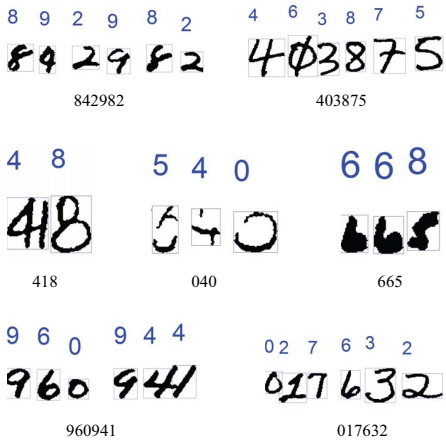

Fig. 4. Examples of mis-recognition of numeral strings.

REFERENCES

[1] T. Ha, M. Zimmermann, and H. Bunke, "Off-line handwritten numeral string recognition by combining segmentation-based and segmentation-free methods," *Pattern Recognition*, vol. 31, no. 3, pp. 257 – 272, 1998.

[2] Y. Chen and J.-F. Wang, "Segmentation of single- or multiple-touching handwritten numeral string using background and foreground analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1304–1317, 2000.

[3] L. Oliveira, R. Sabourin, F. Bortolozzi, and C. Suen, "Automatic recognition of handwritten numerical strings: a recognition and verification strategy," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 11, pp. 1438–1454, 2002.

[4] K. Kim, Y. Chung, J. Kim, and C. Suen, "Recognition of unconstrained handwritten numeral strings using decision value generator," in *Proc. 6th Int. Conf. on Document Analysis and Recognition*, pp. 14–17, 2001.

[5] J. Sadri, C. Suen, and T. Bui, "A genetic framework using contextual knowledge for segmentation and recognition of handwritten numeral strings," *Pattern Recognition*, vol. 40, no. 3, pp. 898 – 919, 2007.

[6] Q.-F. Wang, F. Yin, and C.-L. Liu, "Handwritten Chinese text recognition by integrating multiple contexts," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 34, no. 8, pp. 1469–1481, 2012.

[7] C.-L. Liu, H. Sako, and H. Fujisawa, "Effects of classifier structures and training regimes on integrated segmentation and recognition of handwritten numeral strings," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1395–1407, 2004.

[8] F. Yin, Q.-F. Wang, and C.-L. Liu, "Transcript mapping for handwritten Chinese documents by integrating character recognition model and geometric context," *Pattern Recognition*, vol. 46, no. 10, pp. 2807–2818, 2013.

[9] C. Burges, O. Matan, Y. L. Cun, J. Denker, L. Jackel, C. Stenard, C. Nohl, and J. Ben, "Shortest path segmentation: A method for training a neural network to recognize character strings," in *Proc. International Joint Conference on Neural Networks*, vol. 3, pp. 165–172, 1992.

[10] J. Bromley and J. Denker, "Improving rejection performance on handwritten digits by training with rubbish," *Neural Computation*, vol. 5, no. 3, pp. 367–370, 1993.

[11] J. Liu and P. Gader, "Neural networks with enhanced outlier rejection ability for off-line handwritten word recognition," *Pattern Recognition*, vol. 35, no. 10, pp. 2061–2071, 2002.

[12] H. Xue and V. Govindaraju, "Incorporating contextual character geometry in word recognition," in *Proc. 8th International Workshop on Frontiers in Handwriting Recognition*, pp. 123–127, 2002.

[13] M. Koga, T. Kagehiro, H. Sako, and H. Fujisawa, "Segmentation of Japanese handwritten characters using peripheral feature analysis," in *Proc. 14th Int. Conf. on Pattern Recognition*, vol. 2, pp. 1137–1141, 1998.

[14] T. Fukushima and M. Nakagawa, "Online writing-box-free recognition of handwritten Japanese text considering character size variations," in *Proc. 15th Int. Conf. on Pattern Recognition*, vol. 2, pp. 359–363, 2000.

[15] X.-D. Zhou, J.-L. Yu, C.-L. Liu, T. Nagasaki, and K. Marukawa, "Online handwritten Japanese character string recognition incorporating geometric context," in *Proc. 9th Int. Conf. on Document Analysis and Recognition*, pp. 48–52, 2007.

[16] B. Zhu and M. Nakagawa, "Online handwritten Japanese text recognition by improving segmentation quality," in *Proc. 11th Int. Conf. on Frontiers in Handwriting Recognition*, pp. 521–525, 2008.

[17] C.-L. Liu, "Classifier combination based on confidence transformation," *Pattern Recognition*, vol. 38, no. 1, pp. 11–28, 2005.

[18] Q.-F. Wang, F. Yin, and C.-L. Liu, "Integrating language model in handwritten Chinese text recognition," in *Proc. 10th Int. Conf. on Document Analysis and Recognition*, pp. 1036–1040, 2009.

[19] V. Lavrenko, T. Rath, and R. Manmatha, "Holistic word recognition for handwritten historical documents," in *Proc. International Workshop on Document Image Analysis for Libraries*, pp. 278–287, 2004.

[20] Y. Chung and M. Wong, "High accuracy handwritten character recognition system using contour sequence moments," in *Proc. Int. Conf. on Signal Processing*, vol. 2, pp. 1249–1252, 1998.

[21] C.-L. Liu, H. Sako, and H. Fujisawa, "Handwritten numeral string recognition: Effects of character normalization and feature extraction," *IEICE Trans. Inf. & Syst.*, vol. E88-D, no. 8, pp. 1791–1798, 2005.

[22] R. Casey, "Moment normalization of handprinted characters," *IBM Journal of Research and Development*, vol. 14, no. 5, pp. 548–557, 1970.

[23] C.-L. Liu, "Normalization-cooperated gradient feature extraction for handwritten character recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 29, no. 8, pp. 1465–1469, 2007.

[24] F. Kimura, K. Takashina, S. Tsuruoka, and Y. Miyake, "Modified quadratic discriminant functions and the application to Chinese character recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 9, no. 1, pp. 149–153, 1987.

[25] T.-H. Su, T.-W. Zhang, D.-J. Guan, and H.-J. Huang, "Off-line recognition of realistic Chinese handwriting using segmentation-free strategy," *Pattern Recognition*, vol. 42, no. 1, pp. 167 – 182, 2009.