An Online Handwritten Numerals Segmentation Algorithm Based on Spectral Clustering

Renrong Shao[1], Cheng Chen[2], and Jun Guo*[1]

[1]Computer Center
East China Normal University
3663 Zhong Shan Rd. N., Shanghai, China
[2]iQIYI Innovation Building, No. 365 Linhong Road, Shanghai, China
jguo@cc.ecnu.edu.cn

Abstract. In our previous work, without considering the stroke information, a method based on spectral clustering (SC) for solving handwritten touching numerals segmentation was proposed and obtained very good performance. In this paper, we extend the algorithm to an online system, and propose an improved method where the stroke information is involved. First, the features of the numerals image are extracted by a sliding window. Second, the obtained feature vectors are trained by support vector machine to generate an affinity matrix. Thereafter, the stroke information of original images is used to generate another affinity matrix. Finally, these two affinity matrices are added and trained by SC. Experimental results show that the proposed method can further improve the accuracy of segmentation.

Keywords: stroke information, spectral clustering, handwritten numerals segmentation, affinity matrix, sliding window.

1 Introduction

The segmentation of handwritten touching numerals has been a research focus in the OCR field, because it is the foundation of handwritten numeral string and has great influences on the effect of recognition[12]. In the usual recognition of numerals, it is common practice to divide a numeral string image into single numeral image and then classified by the recognizer and obtain the classification result. It's easy to segment printed numerals. However, for handwritten numerals, due to the connection of handwritten numeral as shown in Fig. 1, these types of numerals will make the segmentation become difficult.

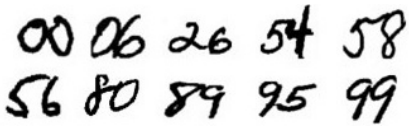


Fig. 1. Different types of touching numerals samples in NIST dataset.

In the traditional algorithms, it is usually based on the contour [3, 8, 18], the skeleton [13, 2, 7], the reservoir [15] or the combination of contour, profile and other morphological features to detect the type of adhesions by connection points [20, 14, 16]. All of these algorithms were proposed to solve some problems of connected numbers, but there are still some drawbacks of these algorithms. With the application of machine learning algorithms in related fields, handwritten single number recognition has become perfect. The recognition accuracy in some excellent recognition systems can reach more than 97% [5], which is close to the level of human. However, the accuracy of handwriting touching numerals recognition have not been able to achieve the desired effect. As far as we know that in ICDAR 2013 (HDRC) competition the single numeral recognition achieved the accuracy of 97.74%[5], and in ICFHR 2014 competition the best accuracy of handwritten numerals string recognition is 85.3%[6]. Both competitions use samples from the CVL database, in which the single numeral samples are extracted from numerals string samples. In recent years, some state-of-the-art algorithms have been proposed to see numerals string as a word, but we still emphasize the importance of segmentation as mentioned in our previous paper[9, 1].

In our previous study, we proposed an approach to extract features by a sliding window which is similar to convolution [1] and use support vector machine (SVM) to predict the affinity matrix of spectral clustering (SC). The matrix predicted in this way need to rely on prior knowledge and large scale samples trained by supervised learning. It only considers the position information of left and right. However, In numerals strings sequence, the left and right information can be replaced by the writing sequence, which means that we can also use strokes sequence instead of left and right information.

Inspired by method in [4], we also can get the gesture and stroke information of our online system. As stroke can be more complete representation of numerals information. Therefore, based on the previous experiments, we use numerals' stroke information to construct a matrix S_m and then add it to the affinity matrix S to obtain an enhanced affinity matrix W^*. Such design can further reflect the internal association of the different numerals, and can more completely response the internal information of the numerals, thus theoretically it can improve the effect of clustering. Experiments show that the improved algorithm has a better effect on the segmentation and further reduces the error rate on the original dataset.

The structure of this article will be introduced as follows: In Sec. 2, the related theoretical algorithms will be introduced. In Sec. 3, we will introduce the improved method and the whole experimental process. In Sec. 4, the experimental results will be shown the comprison effects of our method and previous method on different datasets. Finally, in Sec. 5, we will present our conclusion and future work.

2 Brief of Spectral Clustering

Spectral Clustering is a kind of clustering method based on graph theory. It divides a weighted undirected graph into two or more optimal subgraphs by the distance between nodes in the graph. In graph theory, each graph can be represented as $G = (V, E)$, V is defined as a set of nodes in the graph $\{v_1, v_2...v_n\} \in V$. E is defined as a set of any two nodes connected by the edge, $E(v_i, v_j) \in E$, where $i, j \in n$. Each edge has a weight w_{ij}. As is defined in graph theory, we can construct a matrix of degree of $n * n$ dimensions. However, in practice, the weight w_{ij} of the sample does not actually exist, so it is necessary for us to construct such weights according to certain rules. The common practice in SC is to construct a the affinity matrix S by calculating the distance between any two nodes in the sample. The construction of adjacency matrix W is depended on S. The usual way to calculate distance between two points in space based on their Euclidean distance. The formula is as follows:

$$W_{ij} = S_{ij} = \exp(-\frac{||v_i - v_j||^2}{2\sigma^2}).$$

If a graph's points set is divided into two subsets $\{A, B\}$, then you can define their *cut* as follows:

$$cut(A, B) = \sum\nolimits_{i \in \mathrm{A}, j \in B} w_{ij},$$
$$s.t.\ \ A \cup B = V,\ A \cap B = \emptyset.$$

In spectral clustering, the $cut(A, B)$ gives a measure of association between the two subgraphs. If the graph is divided in an optimal way, then the corresponding *cut* value often is a local minimum. However, in practical problems, a graph with K subgraphs: A_1,A_2,...A_k, there exists multiple local minimum and we do not know which one to use. If we just simply find the global minimum, we will get some unbalanced results, which often leads to isolated points in the cut. In order to avoid the poor result caused by the minimum cut, we need to make a scale to limit each subgraph. Therefore, only by solving the optimization problem: $\min_A Ncut(A_i, A_2, ...A_k)$, we can get the optimal partition of a graph. Although this optimization problem is NP-complete, we still can solve it by introducing slack variables as follow:

$$(D - W)y = \lambda Dy,$$

where $D_{\mathrm{i}} = \sum_{j=1}^{n} w_{ij}$. From this formula, we know that the key to SC is to construct affinity matrix W and its degree matrix D. Based on this theory, our previous method was proposed to construct such a affinity matrix based on pixel features, and now we further strengthen such matrices by adding stroke information.

3 The Proposed Method

In our method, Firstly, sliding window was used to extract features. The design of sliding window refers to the approach of convolution neural network (CNN),

which makes the feature extraction not only limited to the neighborhood calculation of the original image pixels, but can take a deeper step to utilize the left and right position information that hidden inside the numerals. So the relationship between the image pixels can be excavated to make the meaning of the image feature richer. SVM which is a classical machine learning algorithm has a good theoretical basis and extensive application. In addition, we add the stroke information to the sample, and construct the affinity matrix by the stroke information, which can completely reflect the internal correlation of numerals to make the overall effect of the experiment further improved.

3.1 Feature Extraction by Sliding Window

The key to the application of machine learning in field of computer vision (CV) is to completely extract effective features of the image and represent the feature data correctly. In our method, we refer to the convolution operation and use the convolution-like sliding window to extract the features of the samples, which can represent the relation of multi-dimensional data in the image more completely. Each image size in our samples is $32 * 16$ dimensions and sliding window size is $16 * 16$. The main process is to traverse each foreground pixel of the handwritten numerals image. When it encounters the foreground pixel x_i, taking x_i as the center and using the sliding window to extract the pixels around x_i. If the central target pixel is closer to the edge of the image, the part of the sliding window in the periphery of the image is considered as $'0'$ in the region. If the target pixel in the region of window we mark the blank of window as $'1'$. If there is not any target pixel in the window we define the blank of window as $'0'$. So we will get a $16 * 16$ dimensional feature data when a slide window skate over each target pixel. For a sample containing m target pixels, there are $m * 16 * 16$ dimensional feature data. Sliding window to extract the feature data is shown in Fig. 2.

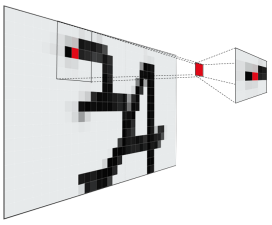


Fig. 2. The $5 * 5$ dimensional sliding window to extract the feature data around the target pixel.

3.2 Construction of Affinity Matrix by SVM

In the part of Sec. 2, the theory shows that using SC to achieve the numerals segmentation based on affinity matrix S. But the key problem is how to construct

this matrix. In a sample, the target pixels is composed of the pixels of the stroke. Usually, the similarity of the SC is to directly calculate the Euclidean distance between any points then to construct the adjacency matrix. Although this method is straightforward and quick, for highly conglutinated numerals, the segmentation effect is still poor. It neglects some of the hidden features and global information of the sample, such as left or right information and stroke information. These features can be used to measure weights between different numerals pixel pairs, so the main task is to extract the features implicit in the foreground pixels and use these hidden features to construct an affinity matrix about the foreground pixels. We can regard the numerals foreground pixels as a one-dimensional matrix, which means it can be defined as $V = \{i_1, i_2, ...i_N\}$ where N represents the number of target pixels, and $i_1, i_2, ...i_N$ represent the target pixel sequence. In the construction of affinity matrix, we can extract the features of $16 * 16$ dimensions around each foreground pixel through the sliding window.

The purpose of the experiment is to get a two-class clustering result by using the SC. Therefore, we hope to get the classification of $'0'$ and $'1'$ which can represent the left-right position of the image through the training of the extracted features. But how to judge such $'0'$ and $'1'$, we can consider it as a dichotomous question. In previous work, we proposed to obtain the value of similarity between two points by SVM prediction [1], So as to obtain a $m * m$ dimensional affinity matrix S of the target pixel. Therefore, the similarity between any two points in a pair of images can be expressed as follows:

$$S_{ij} = P_{svm}(Mbox(i, j)),$$

where $i, j \in N$.

Here P_{svm} represents SVM for predicting the similarity of two points, and $Mbox(i, j)$ represents the use of sliding window $Mbox$ to extract the features of $N*N$ dimensions (N represents arbitrary dimension of matrix, In our experiment is 16) around the center pixel i, j. The correlation between the two pixel pairs is predicted by calculating the features around the center point. This not only limited to the calculation between adjacent pixels, but can make full use of the image features.

In previous work of training models, We normalized the training samples of the data set such that each sample has a binary image of M (M is arbitrary) pixels. The touching numerals in the training sample set are composed by single numeral, and before the synthesis, we marked the left and right numeral as $'L'$ or $'R'$ to represent their location information. Similarly, Using a sliding window to capture a $16 * 16$ dimensional feature of each target pixel, if the central pixel pair has the same label ($'L'$ $'L'$ or $'R'$ $'R'$), the corresponding training sample is marked as positive, otherwise marked as negative.

So if a sample with N foreground pixels, we can get a total of $N * (N - 1)$ data with label, except for the comparison of the pixels themselves (the diagonal of affinity matrix is 0). All these feature data will be used as the input for SVM to train a recognizer model and it will be used to predict the wight of pixel pair in one image.

(a) (b) (c) (d)

Fig. 3. Example of handwritten touching numerals ′23′. (a) is origin sample, the red dots on same side in (b)(d) represent positive samples.the red dots on different side in (c) represent negative samples.

3.3 Construction of Affinity Matrix by Stroke

In practical online application, such as tablet or smart phone, we can get stroke of writing. When the nip of pen touches the pad and leaves the pad, we can record the coordinates of the gliding path. We record each pixel of the numerals to constructs an affinity matrix based on stroke. For the strokes of the handwritten numerals, the pixels in the one stroke are more relevant than the pixels in another strokes. So the correlation coefficient in the same stroke is stronger than other strokes, that is our idea of clustering. The following is our concrete approach. For the one image sample, the stroke of the target pixels can be defined as a set R. If the numeral is composed by two strokes, we define $\{R_1, R_2\} \in R$ for it. Since the numeral ′4′ and ′5′ are both composed of two strokes, so in practical writing connected numerals will appear three or four strokes. The composition of the three stroke information can be represented as $\{R_1, R_2, R_3\} \in R$. For handwritten touching numerals was composed by two numbers, there will not be more than four stroke information in the combination. Maybe someone questions that how we deal with the cluster of numerals more than two strokes. Here, we solve this problem by add the matrix based on our previous method. Because the previous method has divided the numeral string into two categories, and now the numeral with three or four stokes will still be divided into two categories. We extract the target pixel to be a one-dimensional matrix $A[n]$, where n is the length of a one-dimensional matrix. n is also the size of the target pixel while corresponds to the length of affinity matrix. In order to facilitate the calculation, we set two cursors i, j, where i represents the current pixel, and j represents the other target pixels. The calculation of algorithm as follows:

$$
\begin{aligned}
S_m &= A_{ij} \odot {A_{ji}}^T, \\
s.t.&\quad i, j < n. \\
W^* &= S + S_m,
\end{aligned}
$$

Here S_m represents affinity matrix of the strokes. $\odot$ represents calculation of XNOR gate or equivalence gate. From the above calculation of the matrix, we will get a affinity matrix of strokes. The matrix S_m contains the relevant information of the stroke and adds it to the affinity matrix predicted by the SVM before, so as to obtain a new affinity matrix W^*. The matrix W^* is used as the input of SC algorithm. Experiments show that the method with stroke information added performs better than the previous approach. The segmentation effect is

improved significantly. The effect of segmentation comparison is shown as below Fig. 4.

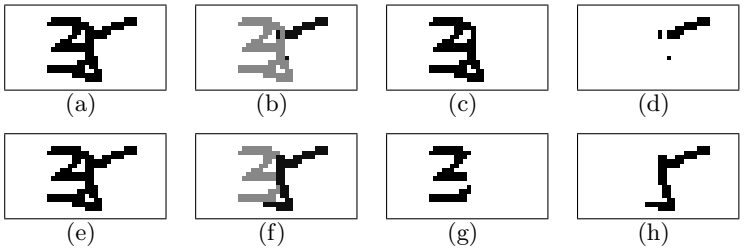


Fig. 4. Example of handwritten numerals ′35′. (a)-(d) is the segmentation effect of handwritten numerals by SC algorithm in previous work. (e)-(h) is the segmentation effect of handwritten touching numerals by SC algorithm with stroke information in this paper.

3.4 Segmentation Verification

From the above, the numerals will be split into two single characters. The following work is to identify the numeral of segmentation image and calculate the accuracy of the segmentation. The common practice is to use the origin single numeral images with prior knowledge as training samples to obtain a recognizer and calculate the accuracy through cross-validation. As our previous test data set was synthesized by single numeral images, so we can train these single numeral data set to get a common recognizer for our segmentation numerals. We choose SVM to do the classification and cross-validation, this part also can be replaced by multi-layer perception (MLP) or back propagation (BP). All the origin image was classified to ′0′ to ′9′ and marked the label, then the training samples from ′0′ to ′9′ were cross-validated by SVM to get the segmentation accuracy. This approach reflects the advantages of the idea of segmentation identification. If we recognized the numerals from ′00′ to ′99′ directly, we need to make 100 classifiers of these numerals respectively. If we do like that, it will increases the complexity of classifier and reduces the reusability. In order to make the whole system more reliable, the accuracy of each numeral' recognizer over 99% on our original sample will be accepted.

4 Experimental Results

The work in this paper is based on the real part of OCR products. Our experimental data was collected from a such a tablet, and data set was supplemented before the experiment. We collected 8,654 single numeral added to the previous data set to train the recognizer. We also collected 9,896 handwritten connected

numerals with stroke information and the size of each sample is $32 * 16$. The number of each numerals is about 100 ranging from $'00'$ to $'99'$. In the experiment, the sample picture and the binary pixel text with the stroke information are used as input for the experiment. The whole process of collecting spent a lot of time and resources and the purpose is to ensure the quantity and quality of the sample, because this part of work has an important role on the following feature extraction, and affects the entire experiment results.

In the part of feature extraction, we still use the previous experiment method. In the part of algorithm, we added an affinity matrix based on the stroke information. In the part of recognition and verification, the data set is equally divided into 10 subsets to use 10-fold cross-validation. For each fold, we use nine of the subsets to build the training sets and the rest of subsets is used as test data for verification. All the algorithms are mainly implemented by python and MATLAB. The prediction of affinity matrix and cross-validation part use the open source framework LIBSVM. In our experiment, we only focus on the accuracy of segmentation, so there is no comparison of calculation time, nor the specific operating conditions associated with specific experiments.

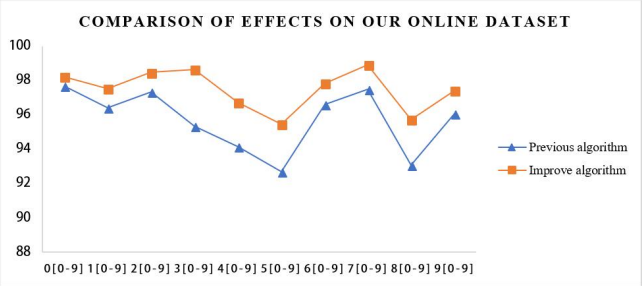
The experiment is tested on our collected online data set and the standard data set (NIST SD-19)[10]. In the our dataset, the previous method without adding the strokes information to obtains an accuracy of 95.68%. Afterwards, when we add the strokes information as input, the whole experiment result

Table 1. Comparison of segmentation effects on our data set and NIST SD-19 data set.

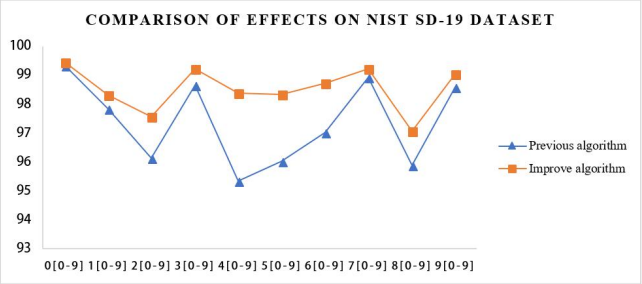
Data sets	Our Online Data Set		NIST SD-19	
Type of Sample	Previous algorithm (%)	Improved algorithm (%)	Previous algorithm (%)	Improved algorithm (%)
0[0-9]	97.68	98.22	99.33	99.46
1[0-9]	96.43	97.54	97.84	98.31
2[0-9]	97.32	98.47	96.15	97.56
3[0-9]	95.34	98.64	98.65	99.23
4[0-9]	94.16	96.73	95.34	98.39
5[0-9]	92.67	95.44	96.02	98.34
6[0-9]	96.59	97.83	97.03	98.72
7[0-9]	97.48	98.92	98.92	99.22
8[0-9]	93.08	95.71	95.88	97.06
9[0-9]	96.08	97.43	98.58	99.05

Table 2. Comparison of different segmentation algorithms using NIST SD-19 data set.

Ref.	Primitives	Ligatures	Pre-class	≥ 2	OverSeg	Approach	Size	Perf.(%)
Pre. approach[1]	Affinity Matrix	Yes	Yes	No	Yes	Seg-Rec-Seg	2000	97.65
[21]	Contour,Concavities	No	No	Yes	No	Seg-then-Rec	3287	94.8
[11]	Contour,Concavities	No	Yes	No	No	Rec-Based	3500	92.5
[12]	Contour	No	Yes	Yes	Yes	Rec-Based	3359	97.72
[19]	Skeleton	Yes	No	No	No	Seg-then-Rec	2000	88.7
[17]	Skeleton	Yes	Yes	Yes	Yes	Rec-Based	5000	96.5
New approach	Improved Affinity Matrix	Yes	Yes	No	Yes	Seg-then-Rec	2000	98.53



(a) the effects on our online data set



(b) the effects on NIST data set

Fig. 5. Comparison of experimental results on different data sets

has been improved to 97.49%. In order to fully demonstrate the credibility and accuracy of our algorithm, we also validate it on another common data set, NIST SD-19. Since NIST SD-19 is an offline data set, it does not provide strokes information. To bring the experimental data closer to our online data set, we perform some preprocessing on the SD-19 data set, manually marking the strokes for samples that have not been divided by the previous method. By comparing with previous method, we found that the segmentation results of the experiment have been significantly improved after adding the stroke information to samples. The segmentation effect of numerals with ′3′ or ′5′ is obviously improved. The whole accuracy of the experimental results on the NIST SD-19 data set can reach 98.53%, which is about 0.9% higher than before. This verifies our previous assumption. The details of comparison of specific experiments is shown in TABLE 1, 2 and Fig.5.

5 Conclusion and Outlook

In this paper, we propose an algorithm that merging the stroke information to the construction of affinity matrix, which improves the correlation of matrix elements and effect of handwritten touching numerals segmentation algorithm. Different from traditional algorithms, our approach divides the connected numerals image

that is based on all pixels of graph, so that it can overcome the disadvantage of the traditional cutting which only calculates neighborhood of images. In our online system, we make full use of the stroke information to build an affinity matrix to improve the SC. These strokes often are ignored by some applications. Our approach has been tested on our collected data set and NIST SD-19 data set, which achieves a good result. The accuracy of segmentation has all improved to 97%, which further proves that online stroke information has good generalization in numerals segmentation. At present, our experiments are mainly used in online writing systems to solve the problem of double-numerals. In the following work, we will continue to study the segmentation problem of numerals string and characters based on this research.

References

1. Chen, C., Guo, J.: A general approach for handwritten digits segmentation using spectral clustering. In: International Conference on Document Analysis and Recognition (IAPR). pp. 547–552 (2018)
2. Chen, Y.K., Wang, J.F.: Segmentation of Single or Multiple-Touching Handwritten Numeral String Using Background and Foreground Analysis. IEEE Pattern Anal (2000)
3. Congedo, G., Dimauro, G., Impedovo, S., Pirlo, G.: Segmentation of numeric strings. In: International Conference on Document Analysis and Recognition. vol. 2, pp. 1038–1041 (1995)
4. Corr, P.J., Silvestre, G.C., Bleakley, C.J.: Open source dataset and deep learning models for online digit gesture recognition on touchscreens. In: Irish Machine Vision and Image Processing Conference (IMVIP) (2017)
5. Diem, M., Fiel, S., Garz, A., Keglevic, M., Kleber, F., Sablatnig, R.: Icdar 2013 competition on handwritten digit recognition (hdrc 2013). In: International Conference on Document Analysis and Recognition. pp. 1422–1427 (2013)
6. Diem, M., Fiel, S., Kleber, F., Sablatnig, R., Saavedra, J.M., Contreras, D., Barrios, J.M., Oliveira, L.S.: Icfhr 2014 competition on handwritten digit string recognition in challenging datasets (hdsrc 2014). In: International Conference on Frontiers in Handwriting Recognition. pp. 779–784 (2014)
7. Elnagar, A., Alhajj, R.: Segmentation of connected handwritten numeral strings. Pattern Recognition 36(3), 625–634 (2003)
8. Fujisawa, H., Nakano, Y., Kurino, K.: Segmentation methods for character recognition: from segmentation to document structure analysis. Proceedings of the IEEE 80(7), 1079–1092 (1992)
9. Graves, A.: Offline arabic handwriting recognition with multidimensional recurrent neural networks. Advances in Neural Information Processing Systems pp. 549–558 (2008)
10. Grother, P.J.: Nist special database 19 handprinted forms and characters database. National Institute of Standards and Technology (NIST) (1995)
11. Kim, K.K., Jin, H.K., Suen, C.Y.: Segmentation-based recognition of handwritten touching pairs of digits using structural features, vol. 23. Pattern Recognition Letters (2002)
12. Lei, Y., Liu, C.S., Ding, X.Q., Fu, Q.: A recognition based system for segmentation of touching handwritten numeral strings. In: International Workshop on Frontiers in Handwriting Recognition. pp. 294–299. IEEE (2004)

13. Lu, Z., Chi, Z., Siu, W.C., Shi, P.: A background-thinning-based approach for separating and recognizing connected handwritten digit strings. Pattern Recognition 32(6), 921–933 (1999)
14. Oliveira, L.S., Lethelier, E., Bortolozzi, F.: A new approach to segment handwritten digits. In: Proc. of 7th International Workshop on Frontiers in Handwriting Recognition. pp. 577–582 (2000)
15. Pal, U., Choisy, C.: Touching numeral segmentation using water reservoir concept. Pattern Recognition Letters 24(1), 261–272 (2003)
16. Ribas, F.C., Oliveira, L.S., Britto, A.S., Sabourin, R.: Handwritten digit segmentation: a comparative study. International Journal on Document Analysis and Recognition 16(2), 127–137 (2013)
17. Sadri, J., Suen, C.Y., Bui, T.D.: A genetic framework using contextual knowledge for segmentation and recognition of handwritten numeral strings. Pattern Recognition 40(3), 898–919 (2007)
18. Shi, Z., Govindaraju, V.: Segmentation and recognition of connected handwritten numeral strings. Pattern Recognition 30(9), 1501–1504 (1997)
19. Suwa, M., Naoi, S.: Segmentation of handwritten numerals by graph representation. International Workshop on Frontiers in Handwriting Recognition 2, 334–339 (2004)
20. Vellasques, E., Oliveira, L.S., Koerich, A.L., Sabourin, R.: Filtering segmentation cuts for digit string recognition. Pattern Recognition 41(10), 3044–3053 (2008)
21. Yu, D., Yan, H.: Separation of touching handwritten multi-numeral strings based on morphological structural features. Pattern Recognition 34(3), 587–599 (2001)