A General Approach for Handwritten Digits Segmentation Using Spectral Clustering

Cheng Chen and Jun Guo
Computer Center
East China Normal University
Shanghai, China
Email: jguo@cc.ecnu.edu.cn

***Abstract*—In this paper, an approach is proposed to solve a classic segmentation problem on handwritten touching digit pairs using spectral clustering (SC). SC has appeared in many of the state-of-the-art algorithms on image grouping problems recently, while it is a challenging work to build similarities between each pixel. In this paper, support vector machine (SVM) is used to predict the affinity matrix in SC instead of designing a complex function, hence making it a general approach. Different from traditional methods which focus on finding the cutting points or lines, we treat the handwritten string segmentation as a graph partitioning problem, which enables us to separate those digits connected in a very complicated way. We also introduce a 'second-segmentation' to optimize the segmentation result, and find out that the whole algorithm is similar to a multi-layer perception (MLP). Experiment results show that the proposed approach performs satisfactorily with high correct rate while keeping its own advantages as a general approach.**

***Keywords*—segmentation; digit; Spectral Clustering; SVM**

I. Introduction

The segmentation of handwritten digit strings is a widely studied problem. In early OCR systems, digit string recognitions are treated as single digit recognitions by dividing a string into several digits. However, on handwritten samples, it becomes a hard task to separate those touching digits from each other. In the last decade, the application of machine learning became a powerful tool in computer vision and image document recognition. Well designed systems can recognize handwritten single digits with a precision over 97% [1] which is almost at the same level with human. Meanwhile, the problem of recognizing handwritten digit strings is not yet solved. In the ICFHR 2014 competition on handwritten digit string recognition, the best performing result at 85.3% [2] has been reported, compared with 97.74% (99.33% including second guess) in ICDAR 2013 (HDRC) [1] on single digit recognition. Both competitions use samples from the *CVL* database, in which the single digit samples are extracted from digit string samples. It deserves to be mentioned that there are researches like MDLTSM [3], which propose recognitions of strings without segmentations. However, many of those methods suppose that the target string is a word, which means a dictionary is used to improve the final result. From [2] Table I, we can see that the best result from Beijing is based on segmentation, while the result from Pernambuco is without segmentation of which the performance is 7.9% lower.

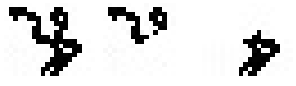
Fig. 1. A sample of handwritten touching digit '33' and the result of K-means segmentation.

In computer vision, there also exist a wide range of problems categorized as *visual grouping*. Previous research has given a series of clustering methods for such a process like K-means, EM algorithm, and spectral clustering (SC). For K-means and EM algorithm, in addition to that we need to know the number of target partitions, the assumptions used in these approaches could be a severe drawback if applied in some specific area like digit string recognition. From Fig. 1 we can see that K-means fails in segmenting handwritten string '33' because the pixels included in one of the two touching digit are far from each other measured with Euclidean distance, while quite a part of pixels from different digits are close enough to become a cluster. One may insist that we can change the assumption and measure those pixels with geodesic distance to improve the performance of K-means and maybe some other clustering methods. This is true. However, the handwritten touching digits are so complex, not only because the digit itself but also the way digits touching each other. And it is hard to tell the number of digits in a touching string, like counting the number of '0' in a string '00000000000000', that is why human put commas in large numbers like '00,000,000,000,000'. Even if all of these problems are properly solved, we are not assured that the new assumption would perform well with other samples like English characters.

There exists a same problem with SC though it has an advantage that the clustering result is insensitive to prior knowledge on the number of target partitions. SC as a graph algorithm, first applied in computer vision in 2000 [4], has become a commonly used framework in visual grouping problems and subspace clustering. The adaptability and flexibility in computing the affinity matrix from data have made this algorithm the state-of-the-art method in many challenging perceptual visual grouping problems. SC works with the assumption that the affinity matrix contains several *blocks*, which means the Ncut (described in section II) between clustered subsets should be optimally small. If given a proper function to compute the pairwise affinity value, SC can separate two intersecting spirals with a precision at 85.9% [5]. But as we have just discussed, it is quite a tricky and challenging task to find such a function, and the function itself is somehow not universal.

In this paper, we try to improve the framework of SC so that we may solve the segmentation problem of handwritten touching digits with a more general approach. We mainly focus on the way to derive affinity matrix from image data. In order to achieve the correct clustering result, the affinity value of point pairs in the same digit should be relatively higher than point pairs from different digits. Therefore, we train an SVM (which could also be replaced by MLP or any other supervised learning models) using subsets of the image as inputs to determine whether the point pair in the subsets is related. Furthermore, we introduce the initial-recognition which make it possible to use additional prior knowledge to improve the result of SC. The entire recognition algorithm consists of three steps:

1. First-segmentation without recognition
2. Initial-recognition based on first-segmentation
3. Second-segmentation based on additional prior knowledge

The rest of this paper is organized as follows. Section II reviews some of the graph theory and SC algorithm briefly. Section III gives a complete definition on our approach and detailed steps for training and predictions. Section IV discusses the experimental results using two different database compared with other existing segmentation algorithm. Finally, the conclusion of this work is stated in section V.

II. Spectral Clustering as a graph algorithm

In graph theory, a weighted undirected graph is noted as $G = (V, E)$, where each edge $(i, j) \in E$ is weighted as $\omega(i, j)$. If the graph is separated into two subsets, then we have a *cut*:

$$cut(A, B) = \sum_{i \in A, j \in B} \omega(i, j), \tag{1}$$

$$where\ A \cup B = V, A \cap B = \phi.$$

In spectral clustering, $\omega(i, j)$ is defined by the similarity between data point i and j, so $cut(A, B)$ gives us a way to measure the similarity between two separated point groups, which means if the graph is partitioned with the optimal way, it will have a local minimum cut.

The problem is that there may be numbers of local minimum cuts in a graph, and we do not know which one to choose. If we simply choose the global minimum cut, we will probably get an unbalanced result that one of the two subsets is too small like single-point-set. Instead of *cut*, spectral clustering uses the so-called *normalized cut (Ncut)*:

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)}, \tag{2}$$

$$where\ assoc(A, V) = \sum_{i \in A, j \in V} \omega(i, j).$$

As we can see, for a given graph, we have $assoc(A, V) + assoc(B, V) = assoc(V, V)$ which is fixed, so if we have several local minimum cuts, the more evenly the graph is partitioned the smaller the corresponding Ncut will be.

Now we can find the optimal segmenting way by solving $min_A Ncut(A, B)\ where\ A \cup B = V, A \cap B = \phi$. Although this optimization problem is NP-complete, it can be rewritten and relaxed as a generalized eigenvalue system [4]:

$$(D - W)\boldsymbol{y} = \lambda D\boldsymbol{y}, \tag{3}$$

$$where\ D(ii) = \sum_j \omega(i, j), W(i, j) = \omega(i, j).$$

Experimentally, if $k \geq 2$ is the number of clusters, the eigenvectors $\{y_2, ..., y_k\}$ corresponding to the k smallest eigenvalues are usually well clustered, so we can apply some simple clustering algorithm like K-means to these $N \times (k-1)$ eigenvectors (provided that $|V| = N$) and the segmentation is done after this step.

As we can see from (3), the result of SC depends upon the only input of W which is also called the affinity matrix. So it is important to find a reasonable way to construct a function for describing the similarity between data points especially in the intersection part. In the last few years, various kinds of methods have been reported like LSA, LRR, SSC and SMMC [5]–[8]. Each of these methods has its own advantage while they are not always interchangeable for the different bias they have. There are also some researches which applied SC on handwritten characters [9]–[11] grouping single-digit examples into different sets like {'4', '6', '8'}, while our work groups pixels in the same multi-digit example.

III. Our approach

We suppose that all the text images as inputs are binary, and each input is treated as a graph G with all the foreground pixels (the text) defined as V. The graph is connected while the weight of each edge remains undefined. As is discussed in section II, the affinity matrix W consisted of the edge weights is the key to SC. Traditional SC is based on local information like Euclidean distance between two data points, which fails in clustering problems like Fig. 1. To avoid such problems, recent studies on SC have improved itself by computing the affinity matrix with more complicated functions. Despite the different theory these works are based on, the spirit of these improvements is to use global information. For example, LRR and SSC suppose that the whole data contains low-rank structures while SMMC is based on low-dimensional manifolds.

A. Affinity Matrix

Unlike natural objects, handwritten characters especially touching strings are not simple geometric figures, so we cannot apply these existing methods directly. However, strings are not randomly distributed pixels. Since characters are written with one pen in order, strings are actually one-dimensional vectors which means the corresponding graph can be noted as:

$$\boldsymbol{V} = \{\boldsymbol{i_1}, ..., \boldsymbol{i_N}\},\ where\ |V| = N, \boldsymbol{i_k} \in V.$$

here $i_1, ..., i_N$ are all foreground pixels, and i_1 is first written, i_2 the second and so on. Though the optical input is a two-dimensional image, it can be reconstructed into a topological order $\boldsymbol{X} = \{\boldsymbol{x_1}, ..., \boldsymbol{x_n}\}$ like the generation problems of handwritten characters [12]. The probability of $\boldsymbol{X}$ can be defined as:

$$P(\boldsymbol{X}) = \prod_{k=1}^{n} P(\boldsymbol{x_k}|\boldsymbol{x_1}, ..., \boldsymbol{x_{k-1}}), \quad (4)$$
$$where\ \boldsymbol{x_k} \in V.$$

As we can see from Fig. 2, in order to get the proper one-dimensional structure, precise n and the exact contents contained in the handwritten string are needed. This leads to a dilemma that the segmentations before recognitions need the prior knowledge of recognition results.

Let us set aside this dilemma and come back to the affinity matrix of SC first. (4) actually gives us a way to describe the relations between data points:

$$\omega(\boldsymbol{i}, \boldsymbol{j}) = max_{\boldsymbol{X}} P(\boldsymbol{X}), \quad (5)$$
$$where\ \boldsymbol{i}, \boldsymbol{j} \in \boldsymbol{X}.$$

Nonetheless, it is hard to solve (5), for the reason that the search space of $\{\boldsymbol{X}\}$ grows exponentially as there exists $\sum_{i=1}^{N-2} i!$ possible topological orders in graph G containing $\boldsymbol{i}, \boldsymbol{j} \in V$. In addition, it is also a challenging problem to define the distribution in (4).

Therefore, a proper estimate of $\omega(\boldsymbol{i}, \boldsymbol{j})$ is preferred here. As we can see from (5), the optimization problem varies with $\{\boldsymbol{i}, \boldsymbol{j}\}$. If we can find a proper way to connect this point pair into one character, then $\omega(\boldsymbol{i}, \boldsymbol{j})$ is high. Conversely, if there exists a way to connect the pair into different character, then $\omega(\boldsymbol{i}, \boldsymbol{j})$ is relatively low. Thus, for each point pair $\{\boldsymbol{i}, \boldsymbol{j}\}$, if we can gather enough examples which have $\boldsymbol{i}, \boldsymbol{j} \in V$, we can use supervised learning method to predict the value of $\omega(\boldsymbol{i}, \boldsymbol{j})$.

In this paper, we train a two-class supported vector machine (SVM) to calculate this affinity value. We suppose that the image examples contain only touching digit pairs, and are all normalized into rectangular images having M pixels (points). Each foreground pixel in training examples is labeled with 'L' or 'R' which means it belongs to the left or right digit. An example is labeled positive for a pixel pair, if the pixel pair has same labels ('L''L' or 'R''R').

In our early experiment we use the whole M pixels as features of SVM, with all foreground pixels valued '1' and '0' for the rest. In the fact that there are N foreground pixels in an example, the affinity matrix is sized as $N \times N$ with its diagonal elements valued 0. Since any of the M pixels in the image example can be foreground pixels, the number of SVMs we need to train is $M * (M - 1)$. This means that a single example appears in $N * (N - 1)$ training sets, while some of the training sets tend to have few examples, like the top-left pixel and bottom-right pixel pair, which make it impossible for the SVM training.

To avoid this problem, we substitute the SVM features with a square fixed-size box of pixels around a foreground pixel

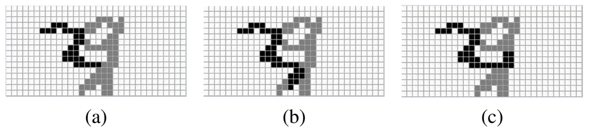
Fig. 2. Examples of time series in handwritten touching digit '39'. (a) n=28, (b) n=36 with high probability assuming the first digit is '3', (c) n=36 with high probability assuming the first digit is '2'.

(which is the central point of the box) as shown in Fig. 3. If the pixel in the box is out of the bound of the original example, it is set with 0. Unlike the former features, this time we train $M_{box} - 1$ SVMs, if the box is sized M_{box}. The reason is that each foreground pixel is taken as the central pixel of a box and then trained or predicted recurrently, so we only have to calculate at most $N - 1$ affinity values for each of them. In addition, the box is smaller than the original example and is packed with more meaningful features, making the SVM training more efficient. Another advantage of this new feature is invariance, because translations of a single digit do not necessarily cause the change of a box.

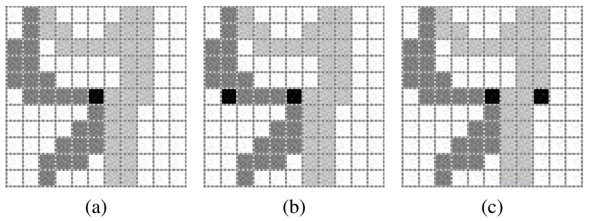
Fig. 3. A box(11*11) of pixels from handwritten touching digit '39'. (a) The black dot is the central point. (b) pairs with (-5,0)(coordinate to the central point)–positive, (c) pairs with (3,0)–negative.

B. Two-step segmentation

As we have discussed on Fig. 2, each digit has its own special one-dimensional structure, hence the need to know

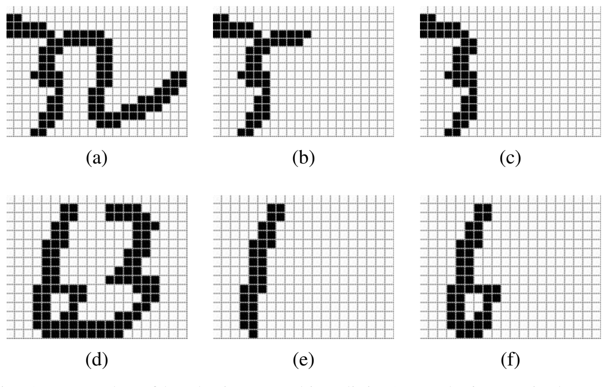
Fig. 4. Examples of handwritten touching digit '32' and '63'. (a) is the used as the input of SC with affinity values predicted by SVMs. (b) is the result of SC using handwritten '5[0-9]' as SVM training sets, (c) uses '3[0-9]' as SVM training sets. (d)-(f) are the same like (a)-(c).

the contents of the string image. From Fig. 4 we can see that handwritten digit pairs can be ambiguous, so if we train the SVMs with different training sets, we will get completely different results from SC while each of the results may be quite acceptable if looked individually.

Since the final goal of most string segmentations is to recognize the characters of it, we cannot expect to know the contents in advance. A simple idea to solve this problem is to implement a two-step segmentation. In the 'first-segmentation', SVMs trained without prior knowledge are used, which means we use mixed training sets, containing all kinds of touching digits from '00' to '99'. However, this leads to the expansion of training sets. To make use of the huge training sets, random sampling is applied, though it will cause the lose of information. In our experiment, the segmentation with mixed training sets does work on a certain degree. From Fig. 5b, we can see that 92.06% of the pixels are clustered correctly, which makes it possible for a rough recognition. The purpose of this rough 'initial-recognition' is to guess the first digit so that only strings started with this digit will be considered next. In the 'second-segmentation', the training sets are restricted to examples like 'X[0-9]', where X is fixed for each clustering result.

As the 'initial-recognition' may give several hypotheses of X with their probabilities, different segmentation results can be produced. In our experiments, the top-1 and top-3 results are given along with their scores corresponded to the outputs of 'initial-recognition'. In an ideal case, the clustering result is optimized in the two-step process as shown in Fig. 5c. On the other hand, a wrong guess may take the result even farther away from the correct segmentation in the worst case.

The entire segmentation algorithm is given as follows:

```
Input: A binary rectangular image with M pixels. Here we
    suppose that the image contains a digit pair.
Output: Top-3 results of segmented image with each fore-
    ground pixels labeled.
 1: for all foreground pixel $p$ in the original image do
 2:     for all foreground pixel $pBox$ in the box around $p$ do
 3:         predict the affinity value between $pBox$ and $p$ using
            SVM with mixed training sets
 4:     end for
 5: end for
 6: carry out spectral clustering
 7: recognize the first digit
 8: for each $i \in [1, 3]$ do
 9:     get a value $x$ from candidate set with highest probability
        according to step 7
10:     repeat step 1 to step 6 with prior knowledge of $x$
11:     adding the clustering result to the result set
12: end for
13: done, return the result set
```

C. Another view on our approach

As a summary, the framework of the whole algorithm is described here in a different way. Just like what is discussed

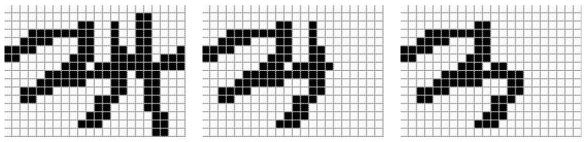

Fig. 5. Examples of handwritten touching digit '34'. (a) used as the input. (b) is the result with mixed SVM training sets. (c) is the result with SVM training sets of '3[0-9]'.

in Subsection *A*, the idea of pixel boxes is actually the same as the design of convolutional neural network (CNN) [13]. The box we used brings the advantages as convolutional kernels do, making the training more efficient while remained accurate and robust. We train SVMs for different prior knowledge as different feature maps, and then SC is applied on the second layer like a full-connect layer. In the case of digit segmentation, we will potentially get 11 different segmentations, so the second-segmentation based on initial-recognition is virtually a softmax of them.

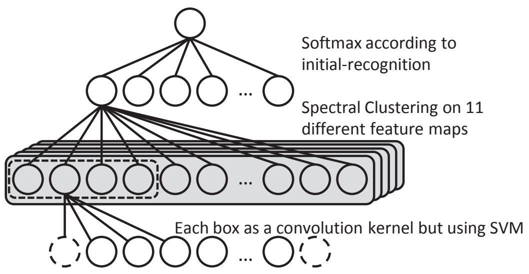
Softmax according to initial-recognition
Spectral Clustering on 11 different feature maps
Each box as a convolution kernel but using SVM

Fig. 6. The framework of our approach as a multi-layer perception.

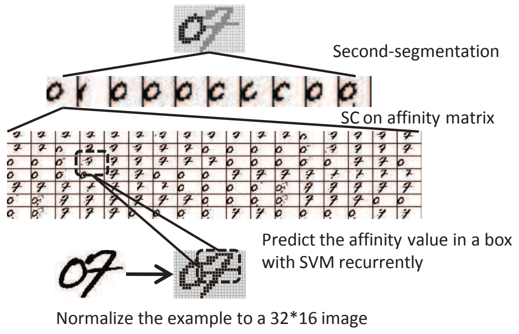
Second-segmentation
SC on affinity matrix
Predict the affinity value in a box with SVM recurrently
Normalize the example to a 32*16 image

Fig. 7. An example of segmentation on touching digit pair '07'.

Fig. 6 illustrates the whole architecture, and from Fig. 7 we can see how it works on a real example step by step. Unlike a normal MLP, our second layer does not have a fixed size, as the number of boxes depends upon the number of foreground pixels. Similarly, not all the feature maps are calculated on each example, since the initial-recognition gives a top-3

TABLE I
COMPARISON OF DIFFERENT SEGMENTATION ALGORITHMS USING NIST SD19 DATABASE.

Ref.	Primitives	Ligatures	Pre-proc	Pre-class	≥ 2	OverSeg	Approach	Size	Perf. (%)
[14]	Contour,Concavities	No	Yes	No	Yes	No	Seg-then-Rec	3287	94.8
[15]	Contour,Concavities	No	Yes	Yes	No	No	Rec-Based	3500	92.5
[16]	Contour	No	Yes	Yes	Yes	Yes	Rec-Based	3359	97.72
[17]	Skeleton	Yes	Yes	No	No	No	Seg-then-Rec	2000	88.7
[18]	Skeleton	Yes	Yes	Yes	Yes	Yes	Rec-Based	5000	96.5
Our approach	Affinity Matrix	Yes	No	Yes	No	Yes	Seg-Rec-Seg	2000	97.65

guess, which means actually four different segmentations are considered each time.

As we know, MLP is successful for the back propagation (BP) algorithm. This also brings us a new idea for training. In current experiments, all the SVMs are trained using boxes of pixels as examples. Since the number of examples is huge, and the number of boxes is usually 50 times bigger, so we have to randomly choose some of them as the training set, which is inefficient since most of the examples selected in this way are not support vectors upon which the performance of SVM depends. In other words, if BP is applied to our approach, it will bring more meaningful training sets, which is the current work of our research.

IV. EXPERIMENTAL RESULTS

The work in this paper is part of a real OCR product, and we have a database of handwritten isolated digits which is collected in previous experiments, containing 3859 examples written by college students. However, in order to train SVMs for affinity value predicting, we need examples of touching digit pair images with each pixel labeled 'L' or 'R'. Clearly, these turn out to be a huge effort for the amount of new examples to collect and the need to manually label them pixel by pixel. In this paper, we use synthetic data for both training and testing set [19]. We create synthetic data by simply connecting two different digit examples together, though there will be some problem like fused example when connecting '7' and '1', we will manually exclude those examples from testing set. We also use another widely used database NIST SD19 to evaluate our approach, and we will compare our results with other researchers.

For all the experiments, the algorithm is implemented in java, python and MATLAB. Libsvm [20] is used for SVM training and predictions. In this work, we concentrate only on the accuracy of the segmentations, thus the computational time will not be compared, and we do not specify the operating conditions.

For our proprietary database, we use 10-fold cross validation. All the isolated digit examples are randomly partitioned into 10 subsets. For each fold, we use nine of the subsets to build the synthetic training sets by sliding two examples (sometimes the same example) horizontally till they touch each other. Here touching means that at least two pixels from the different examples are connected horizontally. The rest subset is built in the same way as validation set. The classifier which initial-recognition uses is trained on the whole database, for we need only to know the precision of segmentations. Each segmentation result is assessed according to the predicted pixel labels. If 95% of the pixels is labeled correctly, then the segmentation is accepted. In our experiment, this means at about most four pixels are labeled incorrectly, which rarely causes a problem in classifying the results. On the opposite, most of the segmentation results with the pixel label accuracy lower than 95% are still recognizable, so the assessment made here is strict and fair. The performance of first-segmentation is 91.59%, and the top-1,top-3 second-segmentation achieves 96.24%,97.31% precision on the whole 14810 validation examples.

Some segmentation algorithms using NIST SD19 are summarized in TABLE I [21]. The result of our approach is added to the last row. The steps of experiment using NIST SD19 database are different in two points:

1. As NIST categorized all the examples by writers, it is preferred to connect the examples from a same writer. Writers from 3600 to 4000 (the ID in NIST database) are considered.

2. To make a fair comparison, a same assessment cretia [21] is applied. A CNN is trained on MNIST as classifier, and all the results must neither be correctly recognized by the classifier while keeping the pixel label accuracy over 90% or keeping the pixel label accuracy over 95% without being recognized. In this way, Type I and Type II errors are eliminated.

Though strings that have more than 2 digits are not considered in this paper, the approach can be applied to such kinds of examples if given the exact number of digits without changing the whole architecture, because SC as a k-class clustering algorithm supports $k \geq 2$. Our algorithm can also be applied in character segmentation, as the features we used are not based on digit.

V. CONCLUSION

In this paper, we developed a segmentation algorithm using spectral clustering and support vector machine. Unlike traditional algorithms, we partitioned the image input pixel by pixel to overcome the limitations of line or point cut segmentation. The whole framework is successfully applied on two different database with precisions over 97%, proving its generality. Several improvements are made like pixel box sampling and second-segmentation, which have significantly optimized the segmentation results.

In our later experiment, we found that BP algorithm can be applied to training, which is our main future work. We are also interested in long digit string segmentation and character segmentation, which we believe can be solved with the same approach proposed.

References

[1] M. Diem, S. Fiel, A. Garz, M. Keglevic, F. Kleber, and R. Sablatnig, "ICDAR 2013 competition on handwritten digit recognition (HDRC 2013)," in *International Conference on Document Analysis and Recognition*, 2013, pp. 1422–1427.

[2] M. Diem, S. Fiel, F. Kleber, and R. Sablatnig, "ICFHR 2014 competition on handwritten digit string recognition in challenging datasets (HDSRC 2014)," in *International Conference on Frontiers in Handwriting Recognition*, 2014, pp. 779–784.

[3] A. Graves, "Offline handwriting recognition with multidimensional recurrent neural networks," in *International Conference on Neural Information Processing Systems*, 2008, pp. 545–552.

[4] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[5] Y. Wang, Y. Jiang, Y. Wu, and Z. H. Zhou, "Spectral clustering on multiple manifolds," *IEEE Transactions on Neural Networks*, vol. 22, no. 7, pp. 1149–61, 2011.

[6] J. Yan and M. Pollefeys, "A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate," *Lecture Notes in Computer Science*, vol. 3954, pp. 94–106, 2006.

[7] G. Liu, Z. Lin, and Y. Yu, "Robust subspace segmentation by low-rank representation," in *International Conference on Machine Learning*, 2010, pp. 663–670.

[8] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 35, no. 11, pp. 2765–2781, 2013.

[9] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means: spectral clustering and normalized cuts," in *Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004, pp. 551–556.

[10] D. Verma and M. Meila, "A comparison of spectral clustering algorithms," 2003.

[11] C. Diao, A. H. Zhang, and B. Wang, "Spectral clustering with local projection distance measurement," *Mathematical Problems in Engineering,2015,(2015-4-19)*, vol. 2015, no. 3, pp. 1–13, 2015.

[12] H. Choi, S. J. Cho, and J. H. Kim, "Generation of handwritten characters with bayesian network based on-line handwriting recognizers," in *International Conference on Document Analysis and Recognition*, 2003, p. 995.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *International Conference on Neural Information Processing Systems*, 2012, pp. 1097–1105.

[14] D. Yu and H. Yan, "Separation of touching handwritten multi-numeral strings based on morphological structural features," *Pattern Recognition*, vol. 34, no. 3, pp. 587–599, 2001.

[15] K. K. Kim, J. H. Kim, and C. Y. Suen, "Segmentation-based recognition of handwritten touching pairs of digits using structural features," *Pattern Recognition Letters*, vol. 23, no. 1-3, pp. 13–24, 2002.

[16] Y. Lei, C. S. Liu, X. Q. Ding, and Q. Fu, "A recognition based system for segmentation of touching handwritten numeral strings," in *International Workshop on Frontiers in Handwriting Recognition*, 2004, pp. 294–299.

[17] M. Suwa and S. Naoi, "Segmentation of handwritten numerals by graph representation," in *International Workshop on Frontiers in Handwriting Recognition*, 2005, pp. 334–339.

[18] J. Sadri, C. Y. Suen, and T. D. 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.

[19] L. S. Oliveira, A. S. J. Britto, and R. Sabourin, "A synthetic database to assess segmentation algorithms," in *Eighth International Conference on Document Analysis and Recognition*, 2005, pp. 207–211.

[20] C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines," *Acm Transactions on Intelligent Systems & Technology*, vol. 2, no. 3, p. 27, 2011.

[21] F. C. Ribas, L. S. Oliveira, A. S. Britto, and R. Sabourin, "Handwritten digit segmentation: a comparative study," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 16, no. 2, pp. 127–137, 2013.