Modeling Segmentation Cuts Using Support Vector Machines

E. Vellasques, L.S. Oliveira, A.S. Britto Jr., A.L. Koerich, R. Sabourin

Modeling Segmentation Cuts Using Support Vector Machines

E. Vellasques, L.S. Oliveira, A.S. Britto Jr., A.L. Koerich, and R. Sabourin
Pontifícia Universidade Católica do Paraná, Curitiba, Brazil
École de Technologie Supérieure, Montreal, Canada
{evellasques,soares,alceu,alekoe}@ppgia.pucpr.br, robert.sabourin@etsmtl.ca

Abstract

In this paper we propose a method to evaluate segmentation cuts for handwritten touching digits. The idea of this method is to work as a filter in segmentation-based recognition systems. These types of systems usually rely on over-segmentation methods, where several segmentation hypotheses are created for each touching group of digits and then assessed by a general-purpose classifier. Through the use of the proposed method, unnecessary segmentation cuts can be identified without any attempt of classification by a general-purpose classifier, reducing the number of paths in a segmentation graph, what can consequently lead to a reduction in computational cost. Concavity analysis is performed in each digit before and after segmentation. The difference of those concavities is used to model the segmentation cuts. SVM is used to classify those segmentation cuts. The preliminary results obtained are very promising as for the segmentation algorithm tested, 67.9% of the unnecessary segmentation cuts were eliminated. Moreover, it was possible to achieve a significant increase in the recognition rate for the general-purpose classifier.

Keywords: Handwritten numerical string recognition, segmentation, Support Vector Machines.

1 Introduction

The recognition of unconstrained handwritten numerals has been a very active area of research. It is composed of several steps, including image acquisition, preprocessing, segmentation, representation, and recognition [3]. Segmentation is a very chalenging task as we need to "split" two or more digits so they can be later recognized by a general-purpose classifier but we also need to know what we are segmenting and that involves some recognition. Early methods [9] used to make heavy use of constraints on document format in order to reduce segmentation complexity.

There are two main tasks in segmentation. The first is connected component detection. Through connected component detection, all the non-touching elements are identified. These elements can be isolated digits, broken parts of digits, delimiters and touching digits. Usually some post-processing is added to this task so broken parts can be grouped. The second and most challenging task is the segmentation of touching digits. A touch between two digits occurs when their foreground pixels merge, creating a bigger connected component. There are two major categories of touching numeral strings, single- and multiple-touching [11]. Figure 1 shows the most common types of touching.

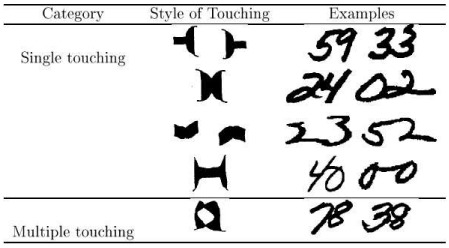

Figure 1. Types of touching between numerals [11].

Casey and Lecolinet [2] proposed a taxonomy for segmentation strategies. According to them, the segmentation strategies can be found in an orthogonal space with three axes, namely: Recognition-based, holistic and dissection. Usually, recognition-based methods make less use of heuristics. However, they usually generate too many segmentation hypotheses and it can become a bottleneck as each digit of these hypotheses have to be later verified by a general-purpose classifier. The dissection methods, otherwise, usually generate less segmentation hypotheses, but depend heavily on heuristics. The literature has many examples that show this taxonomy. In Sadri et al [5], heuristics are heavily used to build segmentation paths. The same occurs in Pal et al [13]. In Fujisawa et al [9] otherwise, heuristics are avoided but for a two-digit string, in some cases three segmentation hypotheses have to be evaluated. In Chen and Wang [11], the use of heuristics is also avoided, but in this case, an average of 7.3 segmentation hypotheses are created for a two-digit touching string [3]. Finally, in Lei et al [1] a recognition-based method is proposed. Figure 2 gives an insight of how many hypotheses should be evaluated by the classifier due to over-segmentation [4].

Heuristics play an important role in segmentation methods. Finding optimal segmentation cuts in a straightforward and general manner is something very difficult

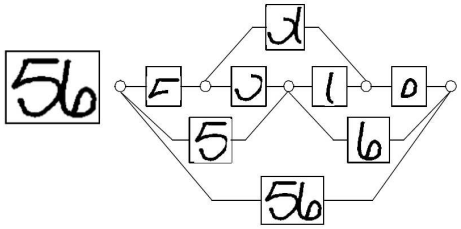

Figure 2. Segmentation paths for string "56".

due to variability in the location of segmentation cuts. Moreover, each of the different types of touching described in Figure 1 can be easily confused with a part of an isolated digit. This makes the use of heuristics to some extent necessary.

The use of heuristics can be reduced but this usually leads to an increase in the number of segmentation hypotheses. Without heuristics, randomly choosing a single segmentation hypothesis would be somewhat risky as a great number of necessary segmentation cuts would be dismissed and this would affect the string recognition rate. It is very difficult to reduce the use of heuristics without increasing the number of segmentation hypotheses or vice-versa due to the lack of general rules to describe points along with the variability of points location.

Instead of creating a new segmentation method without both heuristics and over-segmentation, our approach will be to choose one of them (recognition-based) and try to mitigate its weakness, that is, try to reduce the number of segmentation hypotheses in a cost effective manner. We will do this through the use of a filter, placed between the segmentation and recognition steps. The purpose of our method is to classify the segmentation hypotheses prior to any attempt of recognition by a general-purpose classifier, what would cause a reduction in the complexity of the graph shown in Figure 2, and could consequently mean a reduction in the computational cost. Since we are dealing with a 2-class problem, we have chosen SVM [12] in order to model the segmentation cuts. An MLP was also used so we could have some parameters for comparison.

There are some methods to evaluate segmentation cuts in the literature. In Sadri et al [5], the overlap and height of the resulting segments are verified. The segmentation hypotheses are then discarded if they exceed a predefined threshold. In Chen and Wang [11] eight structural features about the segmentation path are extracted from 823 optimal segmentation cuts (manually selected). Then a mixture of Gaussian function of size M=20 for these samples is created and used to classify candidate segmentation cuts.

The main difference between our method and the existing methods is that in our method, unnecessary segmentation cuts are modeled through the use of over-segmented digits, rather than trying to model unnecessary cuts through their structural features. Modeling over-segmented isolated digits is more straightforward than trying to model the unnecessariness of a cut. This can be observed by the experimental results of our method, where through this approach it was possible to eliminate up to 67.9% of the unnecessary segmentation hypotheses created by a segmentation method [10] with an increase in the general-purpose classifier recognition performance.

2 Database

2.1 Touching digit

We have used the synthetic database proposed by Oliveira et al [3]. This database contains 273,452 (300 dpi, bi-tonal) handwritten strings touching digits pairs and was generated by connecting 2,000 isolated digits extracted from the NIST SD19 database (hsf_0 subset). According to Wang et al [6], 89% of the touching cases occur in two-digit strings. Moreover, most of the existing segmentation algorithms deal with two-digit strings. The algorithm to connect these digits is based in two simple rules - first, only digits created by the same writer can be connected (writer information is available in the NIST SD19 database) and second, the reference axis along which they slide is the center line. These rules make possible to create samples similar to those created by human writers, avoiding distortions like connecting digits with too much different sizes, different levels of slant or different stroke thickness. In addition, only digits correctly classified by an isolated-digit classifier [4] were used to build this synthetic database, removing the bias on isolated digit classification.

This synthetic database contains 89% of single-touching pairs and 11% of multiple-touching pairs and was split in 5 subsets: base_0 containing 80,000 samples, base_1, base_2 and base_3 containing 40,000 samples each and finally base_4 containing 73,452 samples. One important issue with this database is that due to its synthetic nature, all segmentation points from the ground truth can be considered optimal.

2.2 Isolated digit

An isolated digit database was used in order to create the over-segmented digit model. This database is composed of 20,000 samples from the NIST SD19 database (hsf_0 series).

3 Methodology

The purpose of this method is to classify segmentation cuts of single- or multiple-touching numerical strings of any length. Since there is no relation between the position of a segmentation point and its fitness, the use of structural features was avoided. A segmentation point can appear virtually anywhere, for a given digit class as can be seen in Figure 3 [4]. In addition, it can be observed that there is no relation between point location and digit class.

Another way of facing this problem is, rather than trying to understand what can make a point be considered necessary or unnecessary, try to find if the segmentation cut caused an over-segmentation. The idea is that an unnecessary segmentation cut is one that generates an over-

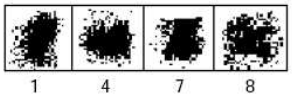

Figure 3. Distribution of segmentation points for isolated digit classes 1, 4, 7 and 8 [4].

segmentation. An over-segmentation can be defined as a cut in an isolated digit. An example of over-segmentation can be seen in Figure 4.

Figure 4. Example of over-segmentation.

3.1 Feature set

The innumerous segmentation algorithms presented in the literature, show several strategies and methods to identify candidate segmentation points. Although some of them have been really successful in finding necessary segmentation points, the number of candidate points generated by these algorithms are a good indicator of how hard it can be to find features that identify segmentation cuts with a high discrimination level.

Another approach would be try to find out if any of the segments obtained by a segmentation cut is an over-segmented digit. Oliveira et al [4] proposed a method to detect over-segmented digits. This method is named Multilevel Concavity Analysis (MCA). First of all, each background pixel of the hypothetical over-segmented part and its original touching pair must be labeled with the number of foreground neighbors that it has in the 4-Freeman directions. This is the Initial Concavity Level (ICL) for that pixel. Label images are created for the hypothetical over-segmented part (I_{seg}) and the original touching pair (I_{orig}). An example of an ICL can be seen in Figure 5.

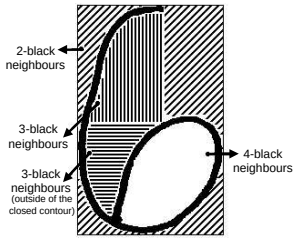

Figure 5. Example of ICL for a digit.

After creating I_{seg} and I_{orig}, a pixel-level comparison of their ICLs is done. As result of this comparison, a new label image, named MCA is created. Each pixel of I_{seg} and I_{orig} is compared, if they have different labels, a specific label is assigned to them in the MCA, indicating that a change in the concavity has occurred. Otherwise the same label is assigned. Foreground pixels from I_{seg} also get a specific label in the MCA image. After creating the MCA, the Contextual Information (CI) for the hypothetical over-segmented part is extracted. The CI of I_{seg} is the ICL of the areas above and below I_{seg}, including all possible foreground pixels. An example of MCA can be seen in Figure 6.

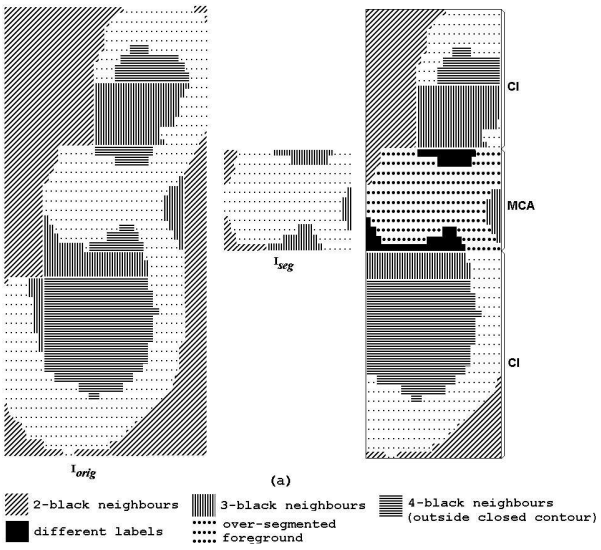

Figure 6. Example of MCA.

There are 7 MCA features that can be extracted from an MCA label image:

1. Number of background pixels surrounded by two black-pixels.
2. Number of background pixels surrounded by three black-pixels.
3. Number of background pixels surrounded by four black-pixels, but not inside a closed loop.
4. Number of background pixels inside a closed loop.
5. Number of background pixels that suffered a change in its concavity level (label).
6. Number of foreground pixels within MCA region.
7. Number of foreground pixels outside MCA region but within extended region.

A zoning scheme is used to extract these features. After creating the MCA label image, the image is divided in 2×3 regions and the 7 MCA features are extracted from each region. The feature vectors of all 6 regions are concatenated in a single feature vector, with 42 features. For each touching-digit pair, two feature vectors are extracted (one for each segment).

The reason of using this scheme is that an over-segmented portion of a given digit, usually do not suffer a so big change on its concavity level, comparing with a correctly segmented digit. Moreover, the changes in the

concavity level for over-segmented digits occur in different locations than the changes in correctly segmented digits, and this behaviour is captured through the use of a zoning scheme.

3.2 Classifiers

Two different classifiers were used, SVM and MLP. The purpose of assessing this method with more than one classifier is to separate the limitations on the method from limitations on the classifier itself.

3.2.1 SVM

The idea of using an SVM was due to the nature of the problem as there are only two classes of cuts, necessary and unnecessary. Moreover, SVMs are tolerant to outliers and perform well in high dimensional data.
The concept of SVM was developed by Vapnik [12]. Let us suppose we have a given set of l samples distributed in a $\Re^n$ space, where n is the dimensionality of the sample space, and for each x_i sample there is an associated label $y_i \in \{-1, 1\}$. According to Vapnik, this sample space can be described by an hyperplane separating the samples according to their label ($\{-1, 1\}$). This hyperplane can be modeled using only a few samples from the sample space, namely the support vectors. So training an SVM is simplified to identifying the support vectors within the training samples. After that, a decision function (1) can be used to predict the label for a given unlabeled sample.

$$f(x) = \sum_i \alpha_i y_i K(x, x_i) + b \qquad (1)$$

The function parameters α_i and b are found by quadratic programming, x is the unlabeled sample and x_i is a support vector. The function $K(x, x_i)$ is known as kernel function and maps the sample space to a higher dimension. In this way, samples that are not linearly separable can become linearly separable (in the higher dimensional space). The most common kernel functions are: Linear, Polynomial, Gaussian and Tangent Hyperbolic. These kernels can be seen in Table 1.

Table 1. Most common kernel functions

Kernel Type	Inner Product Kernel
Linear	$K(x, y) = (x \cdot y)$
Polynomial	$K(x, y) = (x \cdot y + 1)^p$
Gaussian	$K(x, y) = e^{-\|x-y\|^2/2\gamma^2}$
Tangent Hyperbolic	$K(x, y) = tanh(\kappa x \cdot y - \delta)$

The kernel chosen was the Gaussian kernel. LIBSVM [7] was used in our experiments. The parameters used were $C = 128$ and $\gamma = 0.5$. These parameters were found through a grid-test that uses cross-fold validation.

3.2.2 Estimating probabilities

One of the problems with SVMs is that they do not work in a probabilistic framework. Several methods to estimate probability for SVMs have been proposed in the literature. Our method makes use of LIBSVM implementation of binary SVM probability estimate. LIBSVM probability estimate is based in the method proposed by Platt [8]. In this method, the empirical data is used to train a sigmoid function. This sigmoid is then used to map the SVM uncalibrated outputs into probabilities, giving a score between 0 and 1 for each classification result.

3.2.3 MLP

The proposed method was also tested using an MLP classifier. The MLP was chosen due to its efficiency to learn large database. This MLP was trained using gradient descent applied to a sum-of-squares error function. The transfer function employed is the familiar sigmoid function. In order to monitor the generalization performance during learning and terminate the algorithm when there is no longer an improvement, we have used the method of validation or early stopping. Such a method takes into account a validation set, which is not used for learning, to measure the generalization performance of the network. During learning, the performance of the network on the training set will continue to improve, but its performance on the validation set will only improve to a point, where the network starts to overfit the training set, that the learning algorithm is terminated. A second stop criterion is the maximum number of epochs. This classifier gives a score between 0 and 1 for each classification result.

3.3 Training

The focus was on finding over-segmented digits. In order to train the model of over-segmented digits, 20,000 samples of isolated digits from the NIST SD19 database were used. The segmentation method proposed by Fenrich [10] was used to create these samples. This generated 10,781 samples of over-segmented digits that were used to train the unnecessary segmentation cut model. Only the smallest segment (the one with the smallest area) of each one of the 10,781 over-segmented digits was used for this purpose. This was done because we noticed that in most cases, the biggest segment in an over-segmented digit is easily confused with a segment obtained by a necessary cut.
The similarity between a segment obtained by a necessary cut and the biggest segment of an over-segmented digit can be seen in Figure 7.
To train the necessary segmentation cut model, the ground-truth information of 5,000 images from the base_0 subset was used. As the points are optimal, the MCAs of both digits were used, that is, two feature vectors were extracted for each sample, totalling 10,000 feature vectors.

4 Experiments

All experiments were conducted using 10,000 samples of two-digit single-touching numerical strings from the base_2 subset. The segmentation algorithm proposed by Fenrich [10] was used to assess the proposed method.

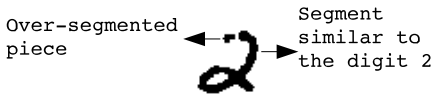

Figure 7. Over-segmented "2" where the biggest segment can be easily confused with a good segment.

This algorithm makes use of contour and profile features in order to find candidate points. The reason to use this algorithm is that it is quite used in the literature. Besides, several segmentation algorithms are based on its features. This algorithm generates in average 3.2 hypotheses for each string (with a standard deviation of ± 1) in the two-digit single-touching string case.
This segmentation method was applied on these 10,000 samples. This segmentation algorithm did not find any segmentation cut for 37 of these 10,000 samples and for this reason these 37 samples were not considered in the filtering experiments. The number of segmentation hypotheses created was 31,305. The criterion to discard a candidate segmentation cut is that the SVM classifier should recognize at least one of the segments as an over-segment. It was also necessary to label these cuts as necessary and unnecessary before doing any experiment. For this purpose, all the 31,305 segmentation hypotheses were submitted to the general-purpose classifier proposed by Oliveira et al [4], which classify the isolated digits and gives a score between 0 and 1 for each classification result. An error rejection option was used. All the cuts were labeled as necessary if the string was correctly classified by this general-purpose classifier and as unnecessary otherwise, resulting in 19,380 unnecessary cuts. Experiments with both, SVM and MLP classifiers were conducted and feature vector elements were normalized between -1 and 1 in the SVM case and between 0 and 1 in the MLP case.

4.1 Performance for optimal segmentation cuts

In the first experiment, the proposed method was tested on the optimal segmentation cuts, provided by the ground-truth, to give a baseline of the system performance. The success rate was of 98.9% in the SVM case and 99.4% in the MLP case. Analyzing the errors occurred in the SVM case, it was possible to notice that more than 50% (0.7% of the 1.1% error rate) of the errors occurred in samples where of the composing digits was either a seven ("7") or a one ("1"). The explanation for this could be the absence (or shortage) of concavity features for these digits. For this reason, they are confused with over-segment as the over-segment usually has not so many concavities as well. But in a real situation the impact of this type of error will be small as touching cases involving them is rare, due to the nature of this type of digit ("1" is usually composed of a single vertical stroke). The use of structural features could help to reduce this type of error, but they also add noise due to their low discrimination power. Some examples of optimal segmentation cuts classified as unnecessary can be seen in Figure 8.

Figure 8. Optimal segmentation cuts classified as unnecessary.

4.2 Performance for unnecessary segmentation cuts

In this experiment, the performance of the classifier to filter unnecessary segmentation cuts was evaluated. There are two important things to be measured - the number of segmentation cuts filtered and the impact on the final string classification results. Firstly, a test without the use of filters was conducted, to give a baseline of the string classification result. As stated before, an MLP classifier was used to classify the strings, using zero-rejection level and error rejection option. Then the filter was added, before the classification. The same criteria was used in order to identify unnecessary cuts, that is, the SVM classifier should recognize at least one of the segments as an over-segment. In the SVM case, 67.9% of the unnecessary segmentation cuts were discarded. When used along with a rejection option, it was possible to obtain an increase in the final recognition rate (by the general-purpose classifier).

Some over-segmented digits are similar to some isolated digits (similar MCA features). For this reason, a few unnecessary cuts were classified as necessary, as can be seen in Figure 9. In Figures 9a, 9b and 9d, the segmentation cut is very close to the optimal segmentation cut found by the segmentation algorithm. In Figure 9c, it is located to the left of the "6" closed loop (it is clearly a segmentation error). But this type of error (false positive) does not affect the general-purpose classifier performance as much as false negative errors do, although Oliveira et al [4] presents some over-segmented digits that lead to error in the general-purpose classification.

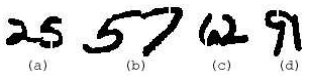

Figure 9. Unnecessary segmentation cuts classified as necessary.

In the MLP case, 59.2% of the unnecessary cuts were discarded, also with an increase in the final recognition rate (by the general-purpose classifier) when compared with using no filter.

5 Conclusion and Discussion

In this article, we proposed a method to classify segmentation cuts. The use of this method is justified by the number of segmentation cuts that an over-segmentation algorithm usually creates. The results obtained are very promising and support the use of this method.

Although the experiments were conducted in two-digit single-touching strings, the use of MCA features makes it possible to expand this method to strings of any length as the process of identifying over-segmented digits in numeral strings is not tied to the string length neither to the number of segmentation cuts. Since an over-segment is found within a string, it should be discarded. The use of structural features otherwise, limit the method to a given string length.

We evaluated the impact of the proposed method in the two-digit string recognition performance. Figure 10 presents the impact of the filters in terms of error rate. In this case None, MLP and SVM stand for the general-purpose classifier without filter, the general-purpose classifier with an MLP filter, and the general-purpose classifier with an SVM filter, respectively.

As we can observe, the SVM filter produces better performance for smaller error rates. Besides, the SVM filter removes more unnecessary cuts, as stated before. On the other hand, MLP filter achieves better performance for smaller rejection rates. But in both cases, the use of the filter brought some improvements in terms of classification performance. This demonstrates the efficiency of the proposed method.

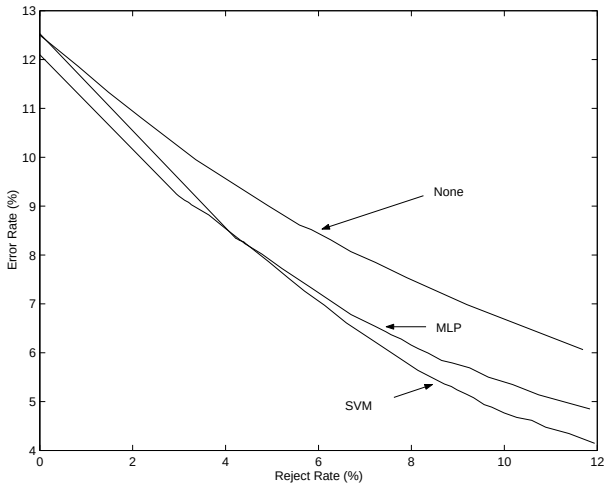

Figure 10. The impact of the filter in terms of error rate versus rejection rate.

In next studies we plan to assess the method in strings of different lengths and containing multiple touches. Finding a way to use structural features in a less constrained manner, that would handle both multiple and single-touching numerical strings of any length is something desired and will also be considered in our future studies (maybe using these features along with the existing MCA features). In addition, we will try to use feature selection in order to find a feature set that maximizes recognition performance. Finally, we will be evaluating the performance of this method in other recognition-based segmentation methods.

Acknowledgment

This research has been supported by The National Council for Scientific and Technological Development (CNPq) grant 476275/2004-0.

6. References

References

[1] Y. Lei, C.S. Liu, X.Q. Ding, and Q. Fu, "A recognition based system for segmentation of touching handwritten numeral strings", In *Proceedings of IFHWR 9*, Tokyo, Japan, 2004.

[2] R.G. Casey and E. Lecolinet, "A survey of methods and strategies in character segmentation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(18):690-706, 1996.

[3] L.S. Oliveira, A.S. Britto, and R. Sabourin, "A synthetic database to assess segmentation algorithms", In *8th International Conference on Document Analysis and Recognition (ICDAR 2005), pages 207-211, Seoul, South Korea, 2005*.

[4] L.S. Oliveira, R. Sabourin, F. Bortolozzi, and C.Y. Suen, "Automatic Recognition of Handwritten Numerical Strings: A Recognition and Verifi cation Strategy", In *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(11):1438-1454, 2002.

[5] J.Sadri, C.Y. Suen, and T.D. Bui, "Automatic segmentation of unconstrained handwritten numeral strings", In *Proceedings of IWFHR 9*, Tokyo, Japan, 2004.

[6] X. Wang, V. Govindaraju, and S. Srihari, "Holistic recognition of touching digits", In *Proceedings of IFHWR 6*, Taejon, South Korea, 1998.

[7] C.C. Chang and C.J. Lin, *LIBSVM: a library for support vector machines*. 2001 Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[8] J. Platt, "Probabilistic Outputs for Support Vector Machines and Comparison to Regularized Likelihood Methods", In A. S. et al, editor, *Advances in Large Margin Classifi ers*, pages 61-74. MIT Press, 1999

[9] H. Fujisawa, Y. Nakano, and K. Kurino, "Segmentation methods for character recognition: from segmentation to document structure analysis", *Proc. of IEEE*, 7(80):1079-1092, 1992.

[10] R. Fenrich, "Segmentation of automatically located handwritten words", In *Proceedings of IFHWR 2*, pages 33-34, Chateau de Bonas, France, 1991.

[11] Y.K. Chen and J.F. Wang, "Segmentation of single- or multiple touching handwritten numeral string using background and foreground analysis", *IEEE Transactions on Pattern Analysis and Machine Intelligence*,11(22):1304-1317,2000.

[12] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995. usually

[13] U. Pal, A. Belaïd, and C. Choisy, "Touching numeral segmentation using water reservoir concept", Pattern Recognition Letters, 24:261-272, 2003.