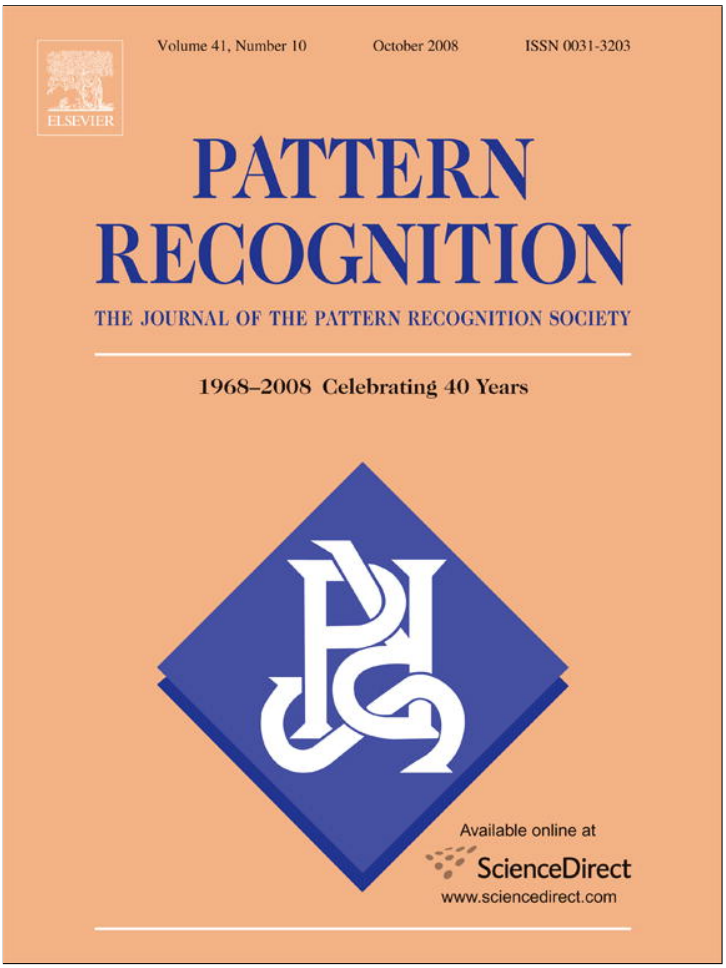

Pattern Recognition 41 (2008) 3044 – 3053

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Filtering segmentation cuts for digit string recognition

E. Vellasques[a], L.S. Oliveira[a,*], A.S. Britto Jr.[a], A.L. Koerich[a], R. Sabourin[b]

[a]Pontifical Catholic University of Parana (PUCPR), Curitiba, Brazil R. Imaculada Conceição, 1155, 80215-901 Curitiba, PR, Brazil
[b]Ecole de Technologie Superieure, 1100 rue Notre Dame Ouest, Montreal, Quebec, Canada

ARTICLE INFO

Article history:
Received 12 January 2007
Received in revised form 20 December 2007
Accepted 13 March 2008

Keywords:
Handwriting recognition
Segmentation
Filtering

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 system. This kind of system 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. The novelty of the proposed methodology lies in the fact that 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. An cost-based approach using ROC (receiver operating characteristics) was deployed to optimize the filter. Experimental results show that the filter can eliminate up to 83% of the unnecessary segmentation hypothesis and increase the overall performance of the system.

1. Introduction

The recognition of unconstrained handwritten numerals is still a very active area of research. It is composed of several steps, including image acquisition, pre-processing, segmentation, representation, and recognition [1]. Segmentation is a very challenging task as we need to "split" two or more digits so that 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 used to make heavy use of constraints on document format in order to reduce segmentation complexity. On first generation OCR's, due to memory limitation, each character had to be scanned individually before its recognition. This approach required pre-scans, where the positions of the characters to be recognized had to be detected first, with the use of reference marks [2].

There are two main tasks in segmentation. The first is connected component detection. Through connected component detection, all the 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 connection 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 [3]. Fig. 1 shows the most common types of touching.

Casey and Lecolinet [1] 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 has 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 [4–6].

In Fujisawa et al. [2], heuristics are avoided but an average of three segmentation points are found for each two-digit string. In Chen and Wang [3], the use of heuristics is also avoided, but in this case, the average of points found for each two-digit touching string is 7.3 [7]. Fig. 2a gives an insight of how many hypotheses should be evaluated by the classifier due to over-segmentation. In this case, suppose that an algorithm proposes SP_0, SP_1, and SP_2 as segmentation cuts, which will divide the image in four segments (C_0, C_1, C_2, and C_3). The set of possible segmentation hypotheses can be represented by a graph, where each segment is represented by a vertex and each segmentation cut is represented by an edge in this graph. SP_1 is the optimal segmentation cut, while SP_0 and SP_2 are unnecessary cuts generated due to the over-segmentation nature of the segmentation algorithm.

A segmentation cut is an "incision" which is applied to a given section of a stroke, which splits the connected component in two parts. Since a connected component may contain more than one character and the connection can occur in more than one region, a common approach is to apply more than one "incision". Each segmentation cut can be turned "on" and "off" accordingly.

* Corresponding author. Tel.: +55 41 3271 1348; fax: +55 41 3271 2121.
E-mail address: soares@ppgia.pucpr.br (L.S. Oliveira).

0031-3203/$30.00 © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2008.03.019

The combination of segmentation cuts and their states ("on" and "off") for a given connected component will determine the set of segmentation hypotheses. A segmentation hypothesis can be seen though as one of the 2^N possible states, where N is the number of segmentation cuts. A segment will be one of the resulting pieces of the connected component after the "incisions" corresponding to a given state have been applied. Here, a segmentation hypothesis can also be seen as the union of such segments. For example, on Fig. 2, if all segmentation points where turned "on", it would result in four segments. This segmentation hypothesis is represented by the central path of the depicted graph. In Fig. 2, the optimal hypothesis is obtained when SP_1 is turned "on" and SP_0 and SP_2 are turned "off". However, turning SP_1 and SP_2 "on" and SP_0 "off" will also result in a high recognition score, in the system proposed by Oliveira et al. [8], leading to recognition error.

The segmentation hypotheses can be generated by several different ways. In the case of Fig. 2a we simulated a classic segmentation algorithm that uses peaks and valleys of the contour to define segmentation cuts.

Finding optimal segmentation cuts in a straightforward and general manner is something very difficult due to variability in the location of segmentation cuts. Furthermore, some results of over-segmentation can be easily confused with an isolated digit (e.g., Fig. 2b where those pieces can be confused with the digits "1" and "0"). In the example depicted in Fig. 2a, the path "510" may produce a higher score than the path "56". This kind of problem makes the use of heuristics to some extent necessary. In this context, a challenging problem lies in reducing 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, in this paper we propose a novel approach to reduce the number of segmentation hypotheses in a cost effective manner. We implemented this through the use of a filter, placed between the segmentation and recognition modules. The purpose of this method is to classify the segmentation cut into necessary or unnecessary prior to any attempt of recognition by a general-purpose classifier, what would cause a reduction in the complexity of the graph shown in Fig. 2a, and could consequently reduce the computational cost. Since we are dealing with a 2-class problem (necessary and unnecessary segmentation cuts), as explained in Ref. [9] we have chosen SVM [10] in order to model the segmentation cuts.

The draft of this concept was first proposed by the authors in Ref. [9] and to the best of our knowledge, there is no similar method in the literature. The main idea behind the proposed method is that unnecessary segmentation cuts are modeled through the use of over-segmented digits, rather than trying to model unnecessary cuts through their structural features. A cost-based approach using ROC (receiver operating characteristics) [11] was deployed to optimize the proposed filter. We have performed experiments using different segmentation algorithms to demonstrate the impacts of such a filter. Experimental results show that the filter can eliminate up to 83% of the unnecessary segmentation hypotheses. We also show that the ROC-based cost mechanism increases the overall performance of the system.

The remaining of this paper is organized as follows: Section 2 describes the baseline system we have used in our experiments and introduces the architectures based on verification and filtering. Section 3 describes the feature set used to train such a filter and also presents the basics about ROC. Section 4 reports the experiments we have performed and Section 5 concludes this work.

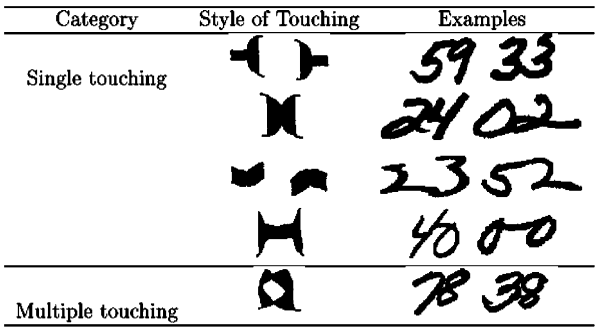

Fig. 1. Types of touching between numerals [3].

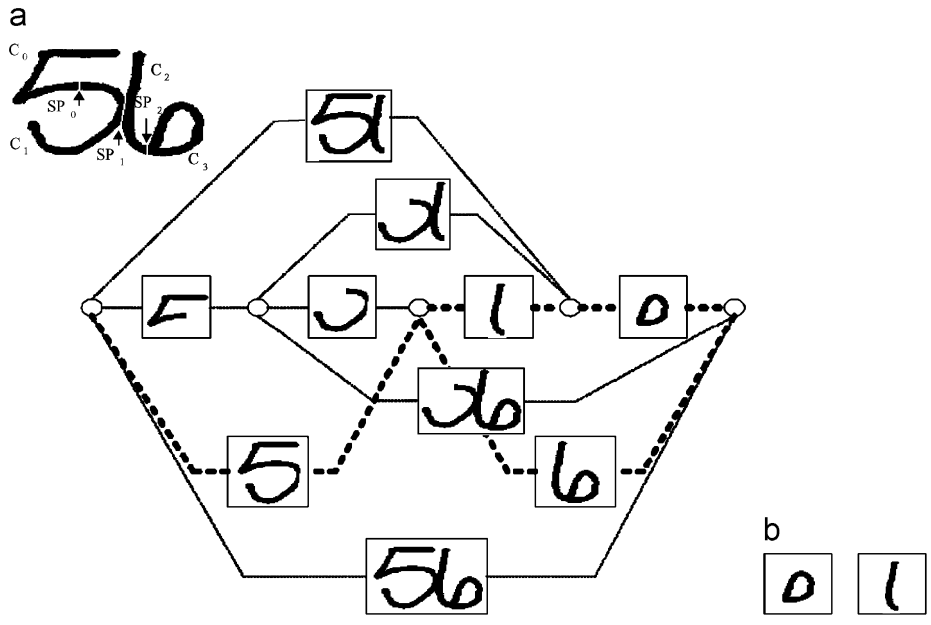

Fig. 2. (a) Segmentation paths for the string "56" and (b) images that can be easily confused with digits "1" and "0".

2. Verification versus filtering

To better assess the impacts of the filter we have used the handwriting recognition system proposed in Ref. [8]. It takes a segmentation-based recognition with an heuristic over-segmentation, where the classifier and verifier are the well-known multilayer perceptron (MLP). The approach combines the outputs from different levels such as segmentation, recognition, and post-processing in a probabilistic model, which allows a sound integration of all knowledge sources used to infer a plausible interpretation. For a complete description of this system, please see Ref. [8]. Fig. 3 depicts the baseline system.

As we can observe from Fig. 3, this system is based on two parallel classification models. The former is the general-purpose classifier (10 classes), which is an MLP fed by a feature vector of 132 components of concavities and contour information. The later is a verifier that tries to detect over-segmentation. This verifier is an MLP, trained to detect over-segmentation. It uses a feature vector called MCA (multi-level concavity analysis), which is composed of 42 features. Later in this paper we will describe this feature vector. The idea of using such a verifier is due to the fact that MLPs are not robust enough to deal with these outliers [12].

Instead of verification, the strategy proposed in this work is based on filtering. Differently of having a parallel classification model, this approach consists of a sequential combination of two experts, called filter and classifier. This concept was introduced by Landgrebe et al. in Ref. [13], where the first stage classifier (detector) attempts to detect target object distributed among a typically poorly sampled, or widely distributed outlier class. Then the second classifier operates on objects selected by the first, and discriminates between sub-target classes. In our work, the first stage classifier is a filter trained to discriminate necessary segmentation points from unnecessary ones while the second one is a general-purpose classifier trained to recognize 10 classes of digits. The filtering strategy is illustrated in Fig. 4. The idea is that we can reduce the computational cost if we detect over-segmentation before calling the general-purpose classifier. To achieve such an objective, a cost-based approach using ROC was used to optimize the filter. We will demonstrate through comprehensive experiments that this strategy can reduce considerably the number of calls to the general-purpose classifier while improving the performance of the recognizer.

3. Implementation

As stated before, the purpose of this method is to classify segmentation cuts of single- or multiple-touching numerical strings. Since there is no relation between the position of a segmentation point and its fitness, the use of structural features was avoided. It has been demonstrated in Ref. [8] that it is very difficult to extract discriminant information from the segmentation cuts. The way we have tackled this problem was, 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 always generates an over-segmented piece. In this way, the filter is designed to discriminate isolated digits (ω_0—necessary points) from over-segmented pieces (ω_1—unnecessary points). An example of these two classes can be seen in Fig. 5.

The idea of using a cost scheme is related to the fact that missing a necessary cut is much worse to the handwriting recognition system than the benefit of correctly identifying an unnecessary cut. In the remaining of this section we describe the feature set used to train the filter and we introduce the concepts of ROC and cost analysis.

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 is a good indicator of how hard it can be to find features that identify segmentation cuts with a high discrimination level. As stated somewhere else, the strategy used in this work lies in identifying over-segmented pieces.

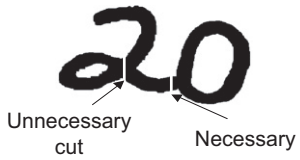

Fig. 5. Example of necessary and unnecessary cuts.

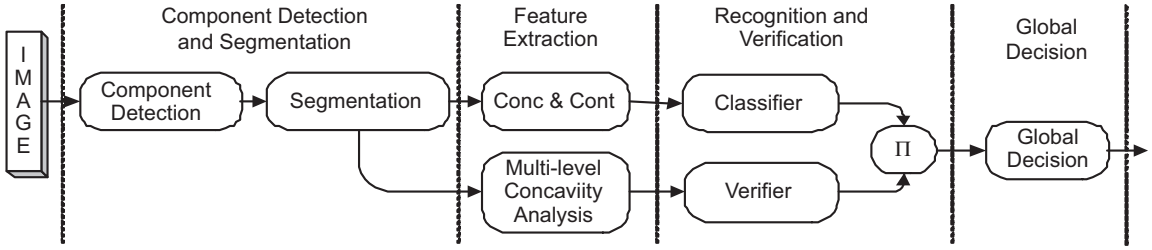

Fig. 3. The block diagram of the baseline system proposed by Oliveira et al. [8]. This system was used in this work.

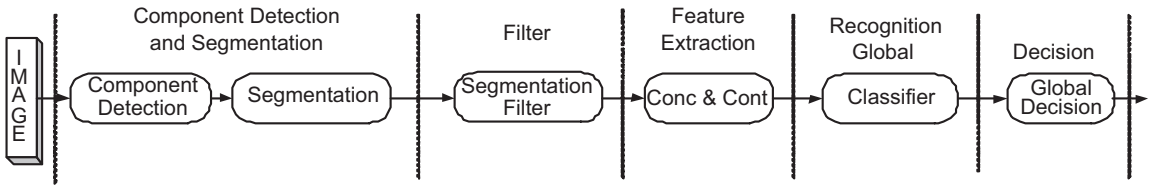

Fig. 4. The block diagram of the proposed strategy.

The premise here is that an over-segmented piece is always generated by an unnecessary segmentation cut. In light of this, we have chosen to use the feature set proposed by Oliveira et al. in Ref. [8], the MCA, which has been successfully applied to discriminate over-segmented pieces from isolated digits.

This feature set is extracted as follows. First of all, each background pixel of the hypothetical over-segmented part and its original touching pair must be labelled with the number of foreground neighbours that it has in the 4-Freeman directions. This is the initial concavity level (ICL) for that pixel. An example of an ICL can be seen in Fig. 6.

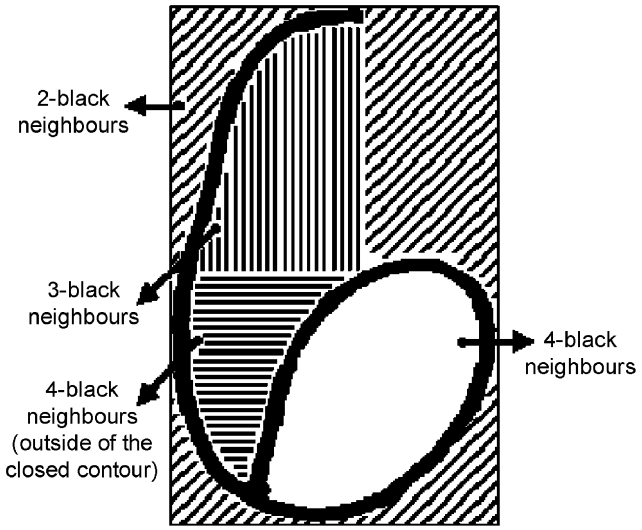

Fig. 6. Example of ICL for a digit.

Label images are created for the hypothetical over-segmented part (I_{seg}, which is one of the segments provided by a given segmentation hypothesis) and the original touching pair (I_{orig}).

After creating I_{seg} and I_{orig}, a pixel-level comparison of their ICLs is done. As a 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. Two examples of MCA can be seen in Fig. 7.

Seven MCA features are extracted from a given image: number of background pixels surrounded by two black-pixels, number of background pixels surrounded by three bla ck-pixels, number of background pixels surrounded by four black-pixels (but not inside a closed loop), number of background pixels inside a closed loop, number of background pixels that suffered a change in its concavity level (label), number of foreground pixels within MCA region, and 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 seven MCA features are extracted from each region. The feature vectors of all six regions are concatenated into a single feature vector, with 42 features. For each segment, one feature vector is extracted.

The reason for using this scheme is that an over-segmented portion of a given digit usually does not suffer such a 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 more than the changes in correctly segmented digits, and this behaviour is captured through the use of a zoning scheme.

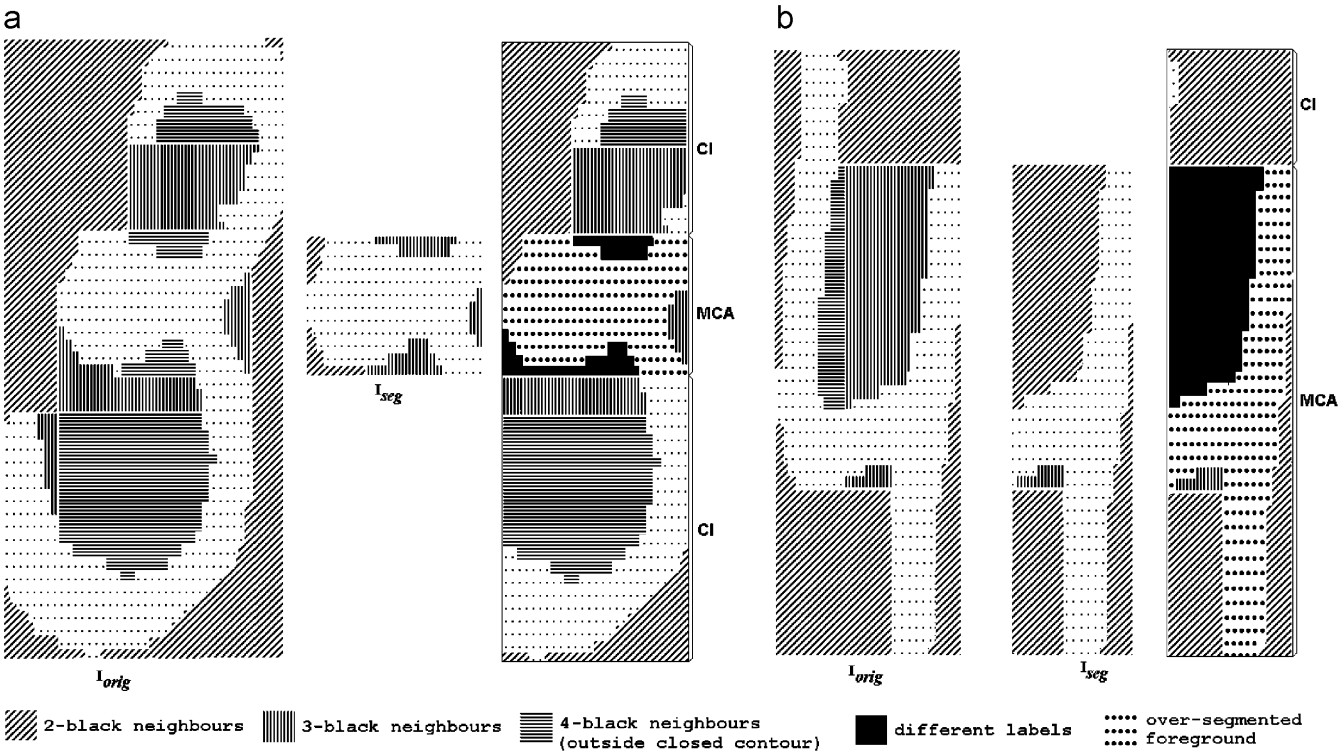

Fig. 7. Example of MCA.

3.2. ROC-based cost scheme

To make this paper self-contained, this section presents a brief introduction of ROC. For a complete reference, please refer to Ref. [14]. Let us consider our 2-class classification problem (ω_0 and ω_1), where each pattern I is mapped to one element of the set $\{\mathbf{p}, \mathbf{n}\}$ of positive and negative class labels. To distinguish between the actual class and the predict class, we use the labels $\{\mathbf{Y}, \mathbf{N}\}$ for the class predictions produced by a model. Given a classifier and an instance, there are four possible outcomes. If the instance is positive and it is classified as positive, it is counted as a true positive (TP); if it is classified as negative, it is counted as a false negative (FN). If the instance is negative and it is classified as negative, it is counted as a true negative (TN); if it is classified as positive, it is counted as a false positive (FP). These measures are presented in the contingency table depicted in Fig. 8.

The numbers along the major diagonal represent the correct decision made while the others represent the confusion between the classes. The following measures can be extracted from this table:

$$\text{TP rate} \approx \frac{\text{Number of true positive instances}}{\text{Number of positive samples}} \quad (1)$$

$$\text{FP rate} \approx \frac{\text{Number of false positive instances}}{\text{Number of negative samples}} \quad (2)$$

Based on that, an ROC can be defined as a two-dimensional graph in which TP rate is plotted on Y-axis and FP rate is plotted on X-axis. An ROC graph depicts relative trade-offs between benefits (TPs) and costs (false positives). Fig. 9a shows an example of class-conditional densities and its corresponding ROC curve.

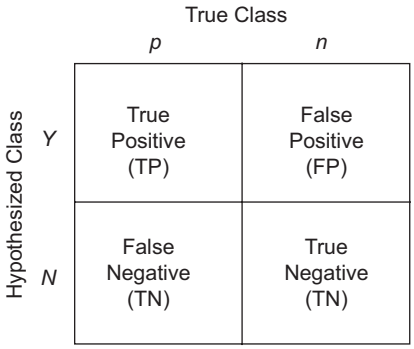

Fig. 8. Contingency table.

Several operational points in ROC space are important to note. The lower left point (0, 0) represents the strategy of never issuing a positive classification. In this case, such a classifier commits no false positive errors but also gains no TPs. The opposite strategy, of unconditionally issuing positive classifications, is represented by the upper right point (1, 1). The point (0, 1) represents perfect classification. Usually, one point in the ROC space is better than another if it is to the northwest (TP rate is higher, FP rate is lower, or both) of the first.

An ROC curve can be a very useful tool either to define a cost scheme [15] or to define a rejection scheme [16]. In our case, we are interested in defining an effective cost scheme since there is nothing to do with a rejected segmentation cut. It must be either classified as necessary (and then recognized by the general-purpose classifier) or unnecessary (and filtered out).

From the filter perspective, the cost of classifying a necessary segmentation cut (ω_0) as unnecessary is CFN (false negative cost), and the cost of classifying an unnecessary segmentation cut (ω_1) as necessary is CTN (true negative cost). The expected classification (EC) cost is given by

$$EC = CFN \times p(\omega_0) \times FN + CTN \times p(\omega_1) \times FP \quad (3)$$

where $p(\omega_0)$ and $p(\omega_1)$ denote the priors for each class. This formulation is adapted from Ref. [16], but here instead of two operating points, only one is used. These two costs will give us a slope, which is defined as follows:

$$m = -\frac{p(\omega_1) \cdot CTN}{p(\omega_0) \cdot CFN} \quad (4)$$

The operational point for a given cost scheme (using these two costs) can be found by searching the point on the ROC curve where the line with slope m intersects the curve. The main idea of the proposed cost scheme is that missing a necessary cut is much worse to the general-purpose recognition performance than the benefit brought by correctly identifying and eliminating an unnecessary cut. If an unnecessary segmentation cut is not detected by the filter, it still can be detected at the recognition level; but if a necessary cut is removed, at the recognition level, it cannot be recovered.

In fact, this strategy minimizes the occurrence of FN, but on the other hand, it also minimizes the occurrence of TN. Later in this paper we will see that indeed, a larger CFN increases the overall recognition performance. For this reason, we tried to "punish" this type of error very hard.

4. Experiments

All experiments were conducted using 15,000 samples of two-digit strings (5000 for training and 10,000 for testing). Besides,

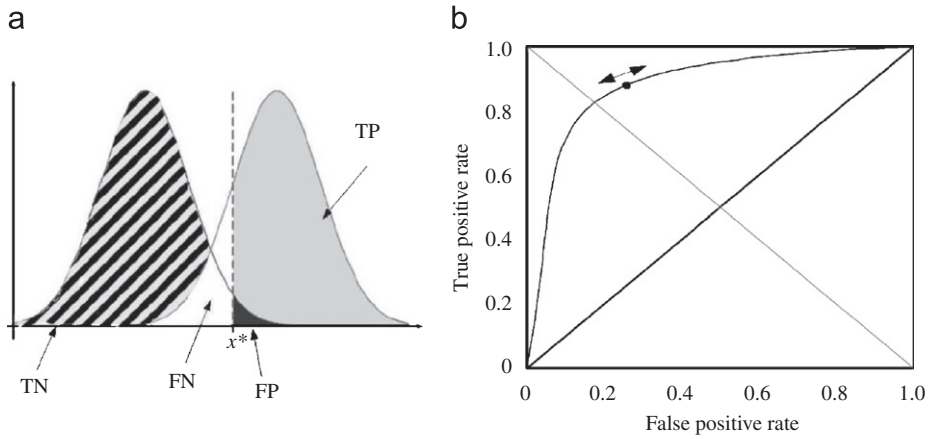

Fig. 9. (a) Class-conditional densities. (b) The corresponding ROC curve and an operating point.

Table 1
Different configurations used in the experimental protocol

Exp.	Description
A	Baseline system without filter and verifier (Fig. 4 without module 2)
B	Baseline system with the verifier and no filter (Fig. 3)
C	Baseline system with filter (Fig. 4). In this configuration, the filter does not use the proposed ROC-based cost scheme
D	Baseline system with filter using ROC-based cost scheme. The goal of this experiment is to highlight the importance of the proposed cost scheme
E	Baseline system with filter using ROC-based cost scheme and verifier. It is a mix of the systems depicted in Figs. 3 and 4.

Fig. 10. Over-segmented "2" where the biggest segment can be easily confused with the digit "2".

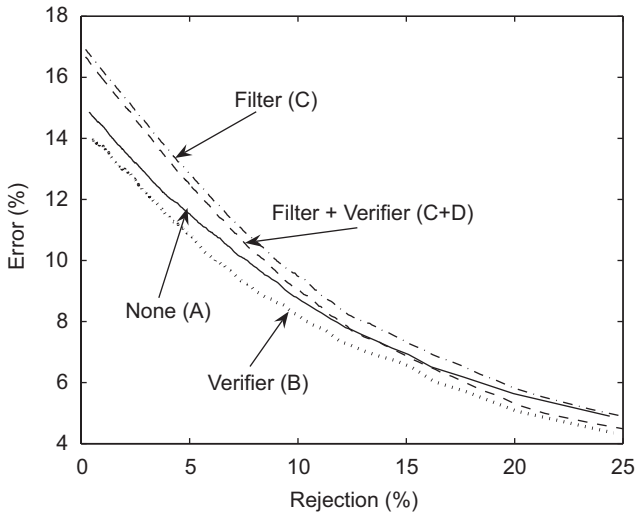

Fig. 11. General-purpose classification performance for No Filter, Filter iso-cost, Verifier, and Filter + Verifier.

10,000 images of isolated digits, extracted from NIST SD19, were considered for training. The touching digit samples were extracted from the synthetic database proposed by Oliveira et al. [7], which contains 273,452 (300 dpi, bi-tonal) handwritten strings touching digit pairs and was generated by connecting 2000 isolated digits extracted from the NIST SD19 database. This database is quite suitable for these experiments because its ground-truth, i.e. the location of the optimal segmentation cuts, is available. By definition, an optimal segmentation cut is the one that correctly separates two characters. Hence, a touching pair can have different optimal segmentation cuts. In this work, though, we assume that the optimal segmentation cut is the one provided by the ground-truth of the database.

Our experimental protocol is two-fold since the impact of the proposed method on both performance and overall computational cost are important in order to evaluate the contribution of the proposed method. To assess the impacts on the computational cost, we have used a measure called TVF (total-number of value features), which was proposed in Ref. [17] and is given by

$$TVF = \sum_{i=1}^{n} m_i x_i \quad (5)$$

where n is the number of classifiers, m_i the number of features of the classifier i and x_i the number of instances classified by the classifier i. This measure is useful to show the burden on the classifier for a given classification task. Since the feature vectors used by both classifier and filter are based on concavities, we assume they have similar cost, and consequently, the cost of extracting such features is not considered. The impact of the filter on the overall performance of the system is demonstrated through the experiments described in Table 1. All these experiments are variations of the systems depicted in Figs. 3 and 4.

With the experimental protocol defined, the first task was devoted to train the filter. As discussed before, the filter was projected to cope with two classes—isolated digits (ω_0) and over-segmented pieces (ω_1). To train ω_1, 10,000 samples of isolated digits extracted from NIST SD19 were used to produce the over-segmented pieces. The segmentation method proposed by Fenrich [18] was applied to these samples generating 10,000 images of over-segmented pieces. Only the smallest over-segmented piece (the one with the smallest area) of each one of the 10,000 over-segmented digits was used for this purpose. This was done because we noticed that in most cases, the biggest segment of an over-segmented digit is easily confused with a segment obtained by a necessary cut. This can be observed in Fig. 10.

To train ω_0, the ground-truth information of 5000 touching digits was used. Since we consider the ground-truth as optimal segmentation cuts, the MCAs of both digits were used, that is, two feature vectors were extracted for each sample, summing up 10,000 feature vectors. In this work, LIBSVM [19] was used to build the filter. The kernel that yielded better results in our experiments was the Gaussian kernel. The parameters used were $C = 128$ and $\gamma = 0.5$. Those parameters were found through a grid-test using 10-fold cross-fold validation.

After training the filter, the next experiments were dedicated to assess its performance on two different segmentation algorithms. The first one was proposed by Fenrich [18] and it is based mostly on information of contour and profile. This algorithm is quite simple and was designed to deal with real problems. Besides, it was the inspiration for several other segmentation algorithms proposed in the literature. The second segmentation algorithm we have chosen is the one proposed by Chen and Wang [3]. Differently from the Fenrich's algorithm, this one uses information of the skeleton as features to generate the segmentation points.

The criterion to discard a candidate segmentation cut is that the filter should assign to ω_1 at least one of the segments of a given segmentation hypothesis (over-segmentation). Basically, each segment of a given segmentation hypothesis (where a segmentation hypothesis corresponds to a path in the graph depicted in Fig. 2) is matched against the filter. If a single segment is classified as an over-segmented piece, the whole segmentation hypothesis is discarded. In this case, one of the possible paths in the graph is eliminated, what will consequently reduce the number of segmentation hypothesis to be classified by the general-purpose classifier.

4.1. Experiments with Fenrich's algorithm

As mentioned before, this algorithm makes use of contour and profile features in order to find candidate points. It generates in average 3.2 segmentation cuts for each two-digit string (with a standard deviation of 1). This segmentation method was applied on 10,000 samples resulting in 31,305 segmentation cuts.

The first experiment carried out consists in comparing the configuration "A", "B", "C", and "C+D" (which is the same as "E") reported in Table 1. In this and all other experiments involving recognition performance, the error versus rejection curve is used. In order to obtain this curve, the error rate for different thresholds on the global

Table 2
Cost schemes used in the experiments

Scheme	CTN	CFN	Threshold	Probability threshold (Negative) (%)
C1	−10	10	1.422	1.21
C2	−10	20	0.430	20.79
C3	−10	30	−1.295	98.21
C4	−10	40	−1.561	99.2
C5	−10	50	−1.607	99.31
C6	−10	60	−1.645	99.38
C7	−10	70	−1.645	99.38
C8	−10	80	−1.889	99.71
C9	−10	90	−2.273	99.99
C10	−10	100	−2.462	99.99
C11	−10	110	−2.949	99.99

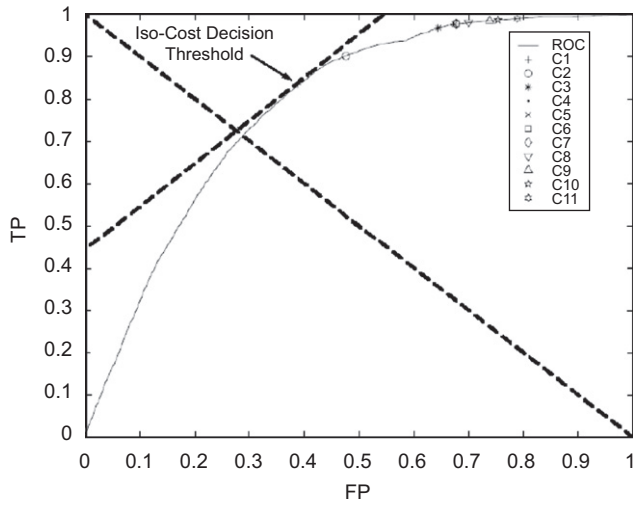

Fig. 12. Operational points for given cost schemes.

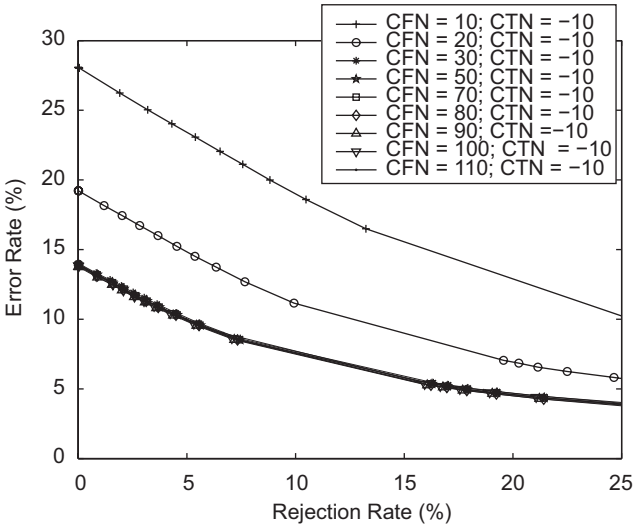

Fig. 13. Performance of the system using the proposed cost schemes.

decision (whole string) was computed. It can be observed from this first experiment (Fig. 11) that the best performance was achieved by system "B". The filter, on the other hand, achieved the worst performance. This is due to the fact that several necessary segmentation cuts were removed. As mentioned previously, this kind of mistake should be avoided, otherwise the general-purpose classifier will not be able to recognize the string. To overcome this kind of problem, we have deployed different cost schemes, which are reported in Table 2.

Eleven different cost schemes, from C1 (iso-cost) to C11 were evaluated. A typical SVM decision function is based on a zero threshold, i.e. if the score (which means the distance of a test vector from the separating hyperplane) is greater than 0, it is classified as positive. Otherwise, it is classified as negative. Choosing a different threshold is the same thing as moving this hyperplane to a new location. Since these scores are unbalanced, we have chosen to convert them into probabilities, using for that the method proposed by Platt [20]. In this scheme, two probabilities are used in order to classify a sample—the probability of being a positive sample and the probability of being a negative sample. The operational points for the given cost schemes can be seen in Fig. 12.

Fig. 13 compares the performance of the filter using the proposed cost schemes reported in Table 2. Since our objective is to assess the performance of the filter rather than the recognition rate of the system, we thought the error/reject trade-offs would be more useful than recognition rates. Another experiment we have done was to test all the cost schemes along with the verifier. The results were very similar to those presented in Fig. 13.

It can be seen that the results of the filter using the cost schemes from C3 to C11 are quite similar. To better compare these

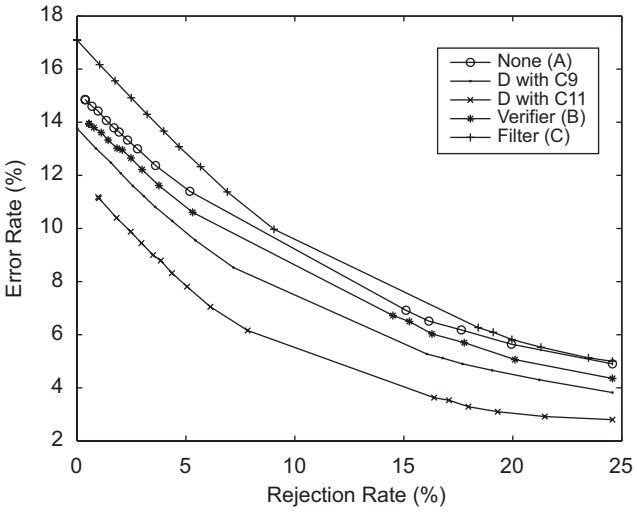

Fig. 14. Comparison among different system configurations.

results, we have applied the best two schemes we found during the experiments (C9 and C11) to the filter. Fig. 14 compares systems "A", "B", "C", and "D" using cost schemes C9 and C11. This figure shows the benefits brought by using the proposed method along with a cost scheme.

Table 3
Impact on computational cost for Fenrich's algorithm

Scheme	# of Hypothesis	# of Filter calls	# of Verifier calls	# of Classifier calls	*TVF* ($\times 10^7$)
No filter	143,533	0	0	527,429	6.96
Verifier	143,533	0	527,429	527,429	9.17
Filter C3	23,519	235,166	0	53,459	**1.69**
Filter C9	37,898	268,710	0	100,122	2.45
Filter C11	49,388	295,100	0	139,524	3.08

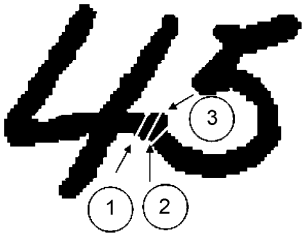

Fig. 15. Similar segmentation cuts generated by Chen and Wang's algorithm.

Besides the improvements brought in terms of performance, the main contribution of the filter is related to the reduction of the computational cost. To demonstrate that, we have used the *TVF* measure presented in Eq. (5). Table 3 shows the impacts of the filter in the computational cost. The columns with value 0 indicates that the module was not used in the test. For example, in the first line *TVF* considers just the calls for the classifier.

As we have seen before (Fig. 13), the performance of the cost schemes from C3 to C11 are very similar, but C3 is much more economic, computationally speaking, than C9 and C11. For this reason we included C3 in the comparison reported in Table 3. We can observe that the number of calls to the classifier could be reduced in about 10 times (from 527,429 to 53,459). In addition, this decrease in the computational cost was obtained along with a reduction of the error rate as depicted in Fig. 13. Other achievement worth of notice is the decrease of 83% in the number of segmentation hypothesis that are evaluated by the general-purpose classifier (from 143,533 to 23,519).

4.2. Experiments with Chen and Wang's algorithm

The segmentation algorithm proposed by Chen and Wang [3] was also used to assess the filter. This algorithm makes use of a thinning algorithm in order to obtain the background and foreground skeletons of the given image. Fork points, end points, and bend points from the skeletons are used as features.

Generally speaking, this algorithm performs better than Fenrich's, but it is more expensive due to the huge number of segmentation cuts generated. Very often, some of the points are quite similar as depicted in Fig. 15. In this figure, we can observe three different segmentation cuts that perfectly segment the image. Point "3" is the ground-truth while points "1" and "2" were generated by the segmentation.

The first question here is: should we retrain the filter or not? We argue that the filter does not need a new training because the shape of over-segmented pieces generated by over-segmentation algorithms are quite similar. To confirm that, we have trained the filter using the over-segmented pieces generated by Chen and Wang's and used it to assess the impacts on the recognition system. The results observed were very similar to those results achieved with the filter trained with over-segmented pieces produced by Fenrich's algorithm.

On the other hand, one thing that does change is the prior for each class. Since this algorithm produces a different number of segmentation cuts, the priors $p(\omega_0)$ and $p(\omega_1)$ are different. Based on that, a new cost analysis was performed, which produced the operating points presented in Fig. 16.

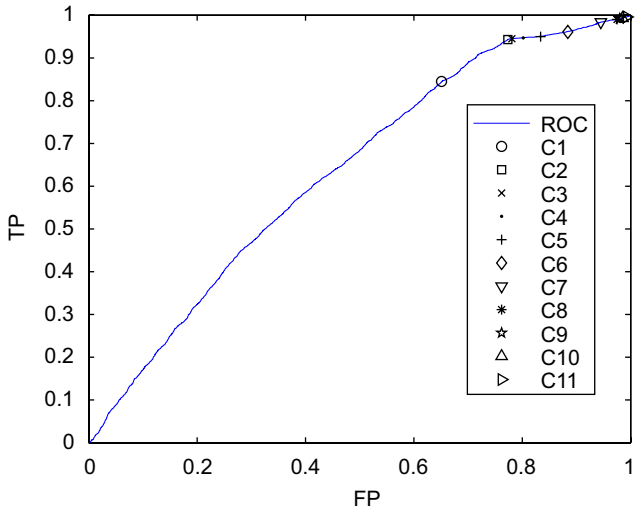

Fig. 16. Operational points for Chen and Wang's algorithm.

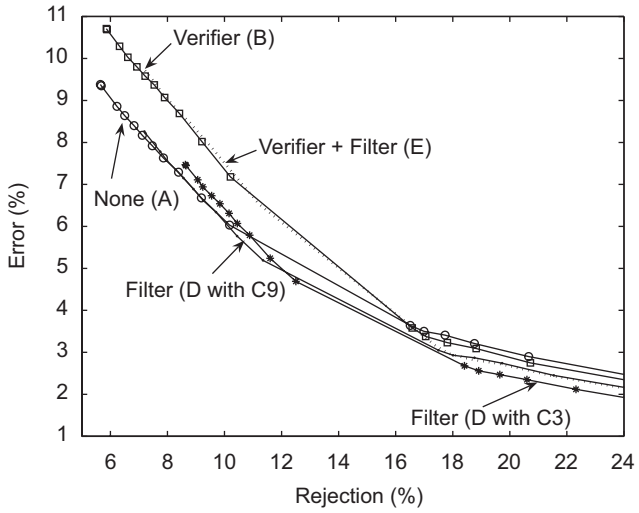

Fig. 17. Impacts of the filter on the recognition system using Chen and Wang's algorithm.

Similar to the previous experiments, here the filter C3 was the one that brought the best results. Fig. 17 shows the impacts of the filter on Chen and Wang's algorithm. As we can observe, the filter C3 was able to bring a slight improvement to the overall performance of the system. However, differently from Fenrich's algorithm, the filter was not able to reduce the computational cost. In such a case, one important aspect that mitigated the impacts of the filter was the huge

Table 4
Impact on computational cost for Chen and Wang's algorithm

Scheme	# of Hypothesis	# of Filter calls	# of Verifier calls	# of Classifier calls	*TVF* ($\times 10^7$)
No filter	235,012	0	0	819,257	10.8
Filter C3	198,414	474,518	0	729,692	12.1
Verifier	235,012	0	498,702	819,257	12.9

Fig. 18. Unnecessary cut not detected by the filter.

number of segmentation points similar to the optimal segmentation cut (see Fig. 15). The consequence is that the filter is called several times and since the segmentation cuts (e.g., points "1" and "2" from Fig. 15) are very similar to the optimal segmentation cut (point "3" from Fig. 15), they are classified as necessary segmentation cuts. Table 4 reports the TVF for Chen and Wang's algorithm.

5. Discussion and conclusion

Although the results obtained show the benefits of the proposed method, there is still room for improvement. As we could observe in the last experiment, the main weakness of the filter is related to the detection of segmentation points similar to the optimal one. In other words, if the segmentation cut is not necessary, but it correctly segments the string in some way, the filter will classify it as necessary. This kind of segmentation point is depicted in Fig. 18, were "0" and "3" where correctly classified in spite of the fact that the segmentation cuts is not the optimal one.

To mitigate this weakness, one strategy would consist in adding some contextual information to the filter. For example, if there is several segmentation cuts classified as necessary in the same region, the filter could keep only one (the one with the highest probability) and discard the remaining. This could alleviate the problem the filter faced in Chen and Wang's algorithm. This will be subject of further studies.

The proposed filter was tested on two different segmentation algorithms. The first one is quite fast and performs well for different touching types. The second is more sophisticated, but it cannot be applied to real application in its present form. Our goal in this paper was to demonstrate the efficiency of the filter but also discuss about its limits.

Although the experiments were conducted in two-digit 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 neither tied to the string length nor 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 limits the method to a given string length.

References

[1] R.G. Casey, E. Lecolinet, A survey of methods and strategies in character segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 18 (7) (1996) 690–706.
[2] H. Fujisawa, Y. Nakano, K. Kurino, Segmentation methods for character recognition: from segmentation to document structure analysis, Proc. IEEE 80 (1992) 1079–1092.
[3] Y.K. Chen, J.F. Wang, Segmentation of single- or multiple touching handwritten numeral string using background and foreground analysis, IEEE Trans. Pattern Anal. Mach. Intell. 22 (11) (2000) 1304–1317.
[4] J. Sadri, C.Y. Suen, T.D. Bui, Automatic segmentation of unconstrained handwritten numeral strings, in: Proceedings of the 9th International Workshop on Frontiers in Handwriting Recognition, 2004, pp. 317–322.
[5] U. Pal, A. Belaid, C. Choisy, Touching numeral segmentation using water reservoir concept, Pattern Recognition Lett. 24 (2003) 261–272.
[6] Y. Lei, C.S. Liu, X.Q. Ding, Q. Fu, A recognition based system for segmentation of touching handwritten numeral strings, in: Proceedings of the 9th International Workshop on Frontiers in Handwriting Recognition, 2004, pp. 294–299.
[7] L.S. Oliveira, A. Britto Jr., R. Sabourin, A synthetic database to assess segmentation algorithms, in: 8th International Conference on Document Analysis and Recognition (ICDAR 2005), Seoul, South Korea, August 29–September 1st 2005, pp. 207–211, IEEE CS Press, Silver Spring, MD.
[8] L.S. Oliveira, R. Sabourin, F. Bortolozzi, C.Y. Suen, Automatic recognition of handwritten numerical strings: a recognition and verification strategy, IEEE Trans. Pattern Anal. Mach. Intell. 24 (11) (2002) 1438–1454.
[9] E. Vellasques, L.S. Oliveira, R. Sabourin, A.S. Britto, A.L. Koerich, Modeling segmentation cuts using support vector machines, in: Proceedings of the 10th International Workshop on Frontiers in Handwriting Recognition, 2006, pp. 41–46.
[10] V. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.
[11] T. Fawcett, An introduction to ROC analysis, Pattern Recognition Lett. 227 (8) (2006) 861–874.
[12] M. Gori, F. Scarselli, Are multilayer perceptrons adequate for pattern recognition and verification?, IEEE Trans. Pattern Anal. Mach. Intell. 20 (11) (1998) 1121–1132.
[13] T.C.W. Landgrebe, D.M.J. Tax, P. Paclik, R.P.W. Duin, C.M. Andrew, A combining strategy for ill-defined problems, in: Fifteenth Annual Symposium of the Pattern Recognition Association of South Africa, 2004, pp. 57–62.
[14] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, second ed., Wiley-Interscience, New York, 2000.
[15] T. Landgrebe, P. Paclikand, D.M.J. Tax, R.P.W. Duin, Optimising two-stage recognition systems, in: Proceedings of 6th Workshop on Multiple Classifier Systems, 2005, pp. 206–215.
[16] F. Tortorella, Reducing the classification cost of support vector classifiers through an roc-based reject rule, Pattern Anal. Appl. 7 (2) (2004) 128–143.
[17] M. Last, H. Bunke, A. Kandel, A feature-based serial approach to classifier combination, Pattern Anal. Appl. 5 (4) (2002) 385–398.
[18] R. Fenrich, Segmentation of automatically located handwritten words, in: Proceedings of the 2nd International Workshop on Frontiers in Handwriting Recognition, 1991, pp. 33–34.
[19] C.C. Chang, C.J. Lin, LIBSVM: a library for support vector machines, 2001, Software available at ⟨http://www.csie.ntu.edu.tw/cjlin/libsvm⟩.
[20] J.C. Platt, Advances in large margin classifiers, in: Probabilistic Outputs for Support Vector Machines and Comparison to Regularized Likelihood Methods, MIT Press, Cambridge, MA, 1999, pp. 61–74.

About the Author—EDUARDO VELLASQUES received the B.S. degree in Informatics from FATEC-SP and M.Sc. degree from the Pontifical Catholic University of Paraná (PUCPR) in 2000 and 2006, respectively. Currently he is a Ph.D. candidate at the Ecole de Technologie Superieure, Montreal, Canada. His interests include pattern recognition and image analysis.

About the Author—LUIZ S. OLIVEIRA received the B.S. degree in Computer Science from UnicenP, Curitiba, PR, Brazil, the M.Sc. degree in electrical engineering and industrial informatics from the Centro Federal de Educacao Tecnologica do Parana (CEFET-PR), Curitiba, PR, Brazil, and Ph.D. degree in Computer Science from Ecole de Technologie Superieure, Universite du Quebec in 1995, 1998, and 2003, respectively. In 2005, he joined the Pontifical Catholic University of Paraná (PUCPR), Curitiba, Brazil, where he is currently an associate professor of computer science. His current interests include Pattern Recognition, Neural Networks, Image Analysis, and Evolutionary Computation.

About the Author—ALCEU S. BRITTO JR received M.Sc. degree in Industrial Informatic from the Federal Center for Technological Education of Parana (Brazil) in 1996, and Ph.D. degree in Computer Science from Pontifical Catholic University of Parana (PUC-PR, Brazil). In 1989, he joined the Computer Science Department of the Ponta Grossa University (Brazil). In 1995, he also joined the Computer Science Department of the Pontifical Catholic University of Paraná (PUCPR), Curitiba, Brazil, where he is currently an associate professor of computer science. His research interests are in the areas of document analysis and handwriting recognition.

About the Author—ALESSANDRO L. KOERICH received the B.Sc. degree in electrical engineering from the Federal University of Santa Catarina (UFSC), Brazil, in 1995, the M.Sc. degree in electrical engineering from the University of Campinas (UNICAMP), Brazil, in 1997, and the Ph.D. degree in automated manufacturing engineering from the ETS Ecole de Technologie Supérieure, Université du Québec, Montréal, Canada, in 2002. In 2003, he joined the Pontifical Catholic University of Paraná (PUCPR), Curitiba, Brazil, where he is currently an associate professor of computer science. His research interests include machine learning, machine vision, and multimedia.

About the Author—ROBERT SABOURIN received B. Ing, M.Sc.A, Ph.D. degrees in Electrical Engineering from the Ecole Polytechnique de Montreal in 1977, 1980 and 1991, respectively. In 1977, he joined the physics department of the Universite de Montreal where he was responsible for the design and development of scientific instrumentation for the Observatoire du Mont Megantic. In 1983, he joined the staff of the Ecole de Technologie Superieure, Universite du Quebec, Montreal, P.Q. Canada, where he is currently a professeur titulaire in the Departement de Genie de la Production Automatisee. In 1995, he joined also the Computer Science Department of the Pontificia Universidade Catolica do Parana (PUC-PR, Curitiba, Brazil) where he was coresponsible since 1998 for the implementation of a Ph.D. program in Applied Informatics. Since 1996, he is a senior member of the Centre for Pattern Recognition and Machine Intelligence (CENPARMI). His research interests are in the areas of handwriting recognition and signature verification for banking and postal applications.