

The recognition of handwritten numeral strings using a two-stage HMM-based method

Alceu de S. Britto Jr^{1,2}, Robert Sabourin^{3,4}, Flavio Bortolozzi¹, Ching Y. Suen⁴

¹ PUC-PR, Pontifícia Universidade Católica do Paraná, R. Imaculada Conceição, 1155, Curitiba (PR) 80215-901, Brazil

² UEPG, Universidade Estadual de Ponta Grossa, Pr. Santos Andrade, Centro, Ponta Grossa (PR) 84100-000, Brazil

³ ETS, Université du Québec, 1100 rue Notre-Dame Ouest, Montréal, Québec H3C 1K3, Canada

⁴ CENPARMI, Concordia University, Suite GM-606, 1455 de Maisonneuve Ouest, Montreal, Quebec H3G 1M8, Canada

Received June 28, 2002 / Revised July 03, 2002

Abstract. In this paper, a two-stage HMM-based recognition method allows us to compensate for the possible loss in terms of recognition performance caused by the necessary trade-off between segmentation and recognition in an implicit segmentation-based strategy. The first stage consists of an implicit segmentation process that takes into account some contextual information to provide multiple segmentation-recognition hypotheses for a given preprocessed string. These hypotheses are verified and re-ranked in a second stage by using an isolated digit classifier. This method enables the use of two sets of features and numeral models: one taking into account both the segmentation and recognition aspects in an implicit segmentation-based strategy, and the other considering just the recognition aspects of isolated digits. These two stages have been shown to be complementary, in the sense that the verification stage compensates for the loss in terms of recognition performance brought about by the necessary tradeoff between segmentation and recognition carried out in the first stage. The experiments on 12,802 handwritten numeral strings of different lengths have shown that the use of a two-stage recognition strategy is a promising idea. The verification stage brought about an average improvement of 9.9% on the string recognition rates. On touching digit pairs, the method achieved a recognition rate of 89.6%.

Keywords: String recognition – Slant normalization – Contextual information – Hidden Markov models – Verification stage

1 Introduction

An important subject of research in the field of document analysis and recognition has been the recognition of numeral strings. The principal motivation is the wide variety of potential applications, such as ZIP codes, bank checks, tax forms and census forms. The challenge is to recognize numeral strings of unknown length which are

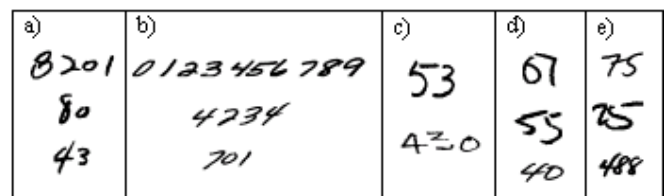


Fig. 1. Difficulties in recognizing numeral strings: **a** size variation; **b** slanted numerals; **c** broken numerals; **d** overlapping numerals; **e** touching numerals

not neatly written. Some possible difficulties contributing to the unsatisfactory performance of many methods for recognizing handwritten numeral strings are shown in Fig. 1.

The more conventional methods attempt to segment the numeral string into individual numerals prior to a recognition step (segmentation-based methods). In these methods, broken numerals represent a significant problem, since their parts need to be grouped to form a meaningful component that is able to represent an individual numeral. Usually, a set of heuristic rules is used to group parts of broken numerals. These rules normally take into account information like height, width and position of adjacent connected components, plus the distance between them. Examples of the effort required to group broken parts of numerals can be found in [10,30]. Another difficult task for segmentation-based methods consists of segmenting touching numerals. Many studies have addressed just this problem [13,33,34]. However, these studies have revealed that segmenting touching numerals without the aid of a recognizer is often unreliable. In addition, the use of heuristics to drive a blind search for digits in strings reduces the accuracy of the segmentation-based methods.

The segmentation-free methods have demonstrated their advantages in dealing with broken and touching numerals. Notice that “segmentation-free” does not mean that no segmentation process is involved. It means rather that the segmentation process is performed with the aid of a numeral recognizer. The recognition can be performed simultaneously with the segmentation process

(implicit segmentation) [12,15,20,21,26], or afterwards, in order to search for the best way to assemble primitive segments to form the string (explicit segmentation) [8,23,24,25].

Usually, in explicit segmentation-based methods an over-segmentation strategy is used to segment a string into primitive segments, which are classified as digits or parts of digits. A further recognition process is used to merge these primitive segments to form a string. In such a method, we still have the string segmentation prior to the recognition step. The difference from segmentation-based methods is that, here, the objective is to obtain primitive segments and not “numeral like” components. However, it is clear that this method still does not represent well a good compromise between segmentation and recognition processes. In fact, a correct recognition often depends on a correct segmentation. By contrast, a correct segmentation also requires a correct recognition. Thus, we can say that these statements have a “chicken and egg” relationship, and they should be approached simultaneously.

An alternative aimed at avoiding the prior segmentation of the string has been the use of implicit segmentation-based methods to integrate segmentation and recognition processes. A promising approach to achieve this has been based on Hidden Markov Models (HMMs). This approach was originally developed in the field of speech recognition [27], where it has been applied with much success. More recently, Bose and Kuo [1] and Elms et al. [6] have shown the benefits of applying it to recognizing printed words. In [26], the method proposed by Elms et al. [6] is adapted for handwritten numeral strings. From these studies, we may conclude that such an approach is a promising way of integrating segmentation and recognition to deal with the difficulties encountered in processing handwritten numeral strings. However, we can also observe some cost attached to this integration, which is an open problem in such an approach. This cost is a loss in recognition performance caused by combining segmentation with recognition. In other words, the problem is that a set of features and models that show promising performance in terms of segmentation usually do not show similar performance in terms of recognition, and vice versa. By contrast, to integrate them it is necessary to define features and numeral models to contemplate both the segmentation and recognition aspects simultaneously. Moreover, a feature set must be extracted from a numeral string image in the same way that it is from an isolated digit image. In summary, the challenge is to find some way to compensate for the loss in recognition performance resulting from the necessary tradeoff between segmentation and recognition carried out in an implicit segmentation-based method.

The proposed method for recognizing handwritten numeral strings provides a way of combining segmentation and recognition taking into account this necessary tradeoff. The method is based on a two-stage recognition strategy that enables the use of two sets of features and numeral models: one taking into account both the segmenta-

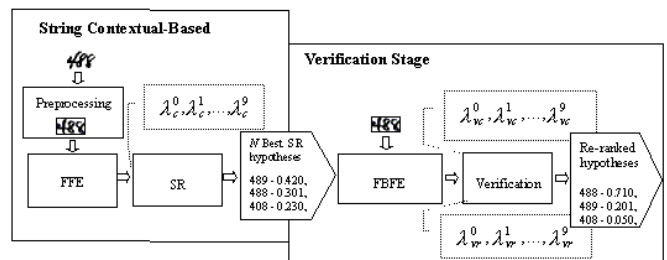


Fig. 2. General overview of the proposed method

tion and recognition aspects in an implicit segmentation-based process, and another considering just the recognition aspects in a further verification process.

This paper consists of five sections. In Sect. 2, the proposed method, which can be categorized as a segmentation-free approach, is described in detail. This description starts with a general overview, then each stage and the corresponding modules are described. Section 3 presents a rigorous experimental protocol for implementing and evaluating the proposed string recognition method. Experiments are performed considering isolated digits, numeral strings of different lengths and touching digit pairs extracted from the NIST SD19 database. Section 4 presents a brief discussion, while Sect. 5 presents the conclusion and future works.

2 Overview of the method

A general overview of the proposed method is presented in Fig. 2. In the SCB (String Contextual-Based) stage, a given numeral string is first preprocessed in order to correct slant, smooth the string contour and calculate the string bounding box. Subsequently, the FFE (Foreground Feature Extraction) module scans the string image from left to right, while a feature vector based on foreground information is calculated for each column in the string bounding box. This vector is mapped to a discrete symbol available in a previously constructed codebook. The output of the FFE module is a sequence of discrete observations representing the entire numeral string. The length of this sequence corresponds to the number of columns in the string bounding box. In the SR (Segmentation-Recognition) module, numeral HMMs (Hidden Markov Models) trained on isolated digits ($\lambda_c^0, \lambda_c^1, \dots, \lambda_c^9$), but considering string contextual information, are matched to the observation sequence provided by the FFE module. The objective is to find the N best segmentation-recognition paths (hypotheses) using an implicit segmentation-based strategy. For each segmentation-recognition hypothesis, the following information is provided: string length (number of digits), segmentation points, recognition result of each string segment and the string recognition score.

The segmentation-recognition hypotheses generated by the SCB stage are re-ranked in the Verification Stage. Basically, this second stage consists of an HMM-based digit classifier trained on isolated digits without taking into account any string contextual information. A new

set of features combines foreground and background information to improve the recognition performance of the numeral HMMs. Moreover, 10 additional numeral HMMs ($\lambda_{vr}^0, \lambda_{vr}^1, \dots, \lambda_{vr}^9$) based on the rows of the numeral images are combined with the column-based models ($\lambda_{vc}^0, \lambda_{vc}^1, \dots, \lambda_{vc}^9$) to ensure an accurate representation of the digit classes.

The verification process starts in the FBF (Foreground/Background Feature Extraction) module, in which the segmentation points provided by the first stage are used to define string segments and calculate their bounding boxes. Then, for a given segment, a feature vector combining foreground and background information is extracted from each column of the segment bounding box. This vector is mapped to a discrete symbol in a previously constructed codebook. A similar process is carried out for the rows of the segment. The output of the FBF module consists of two sequences of discrete observations for each segment: column-based and row-based sequences. In the Verification module, the first step is to select the column ($\lambda_{vc}^0, \lambda_{vc}^1, \dots, \lambda_{vc}^9$) and row ($\lambda_{vr}^0, \lambda_{vr}^1, \dots, \lambda_{vr}^9$) models corresponding to the segment to be verified. This is done by using the recognition result of each string segment provided by the SCB stage. The selected models are scored using the corresponding sequence of observations extracted by the FBF module. The output probabilities of both column and row models are combined by summing their logarithms (logs). A string score is obtained by summing the score of each segment of the numeral string. This string verification score is added to the score of the corresponding segmentation-recognition hypothesis obtained from the SCB stage in order to re-rank it.

2.1 String Contextual-Based (SCB) stage

As previously described, the general objective of this stage is to provide the N best segmentation-recognition paths or hypotheses for a given numeral string. To this end, it is composed of three modules: Preprocessing, Foreground Feature Extraction (FFE) and Segmentation-Recognition (SR). The main characteristic of this stage is the use of an implicit segmentation-based strategy to integrate segmentation and recognition processes. It has been shown that this is a promising strategy for dealing with the string difficulties previously presented, since this allows us to avoid a prior segmentation of the string into digits without the aid of a recognizer. In addition, this stage contains numeral HMMs trained on isolated digits, but considering contextual information (CI) regarding slant and intra-string size variations. The main purpose of using such a CI is to ensure an accurate representation of numeral strings.

2.1.1 Preprocessing module. In this module, the string slant is corrected in order to reduce the script variability. The method proposed in [2] has also shown to be really helpful in alleviating overlapping between adjacent digits

which may interfere with the column-based features extracted from them. The smoothing method described in [31] is used, before and after slant correction, to reduce possible artifacts on the string contour. The last step of this module concerns the calculation of the string bounding box.

2.1.2 Foreground Feature Extraction (FFE) module. In the implicit segmentation strategy of the SCB stage, a preprocessed string image is scanned from left to right, while numeral HMMs are matched to it by means of the Level Building Algorithm [28]. Thus, it is necessary to compute a feature vector as a function of an independent variable through the use of a windowing scheme [6,9,19]. Similar to [6] in our windowing scheme a feature vector is calculated for each column of the digit or string image. With this scheme, the problem of defining the size of a frame or sliding window is avoided, as is the overlap from one frame to the next. In addition, the use of frames which are usually divided into cells, can make the features more dependent on size normalization methods, which may cause distortions in the digit strokes, broken digits or even generate new touching cases. To compensate for not using a zoning scheme based on frames and cells, the relative position of each feature from the top of the digit or string bounding box is computed. This measure is size invariant and it can simulate a zoning scheme.

The FFE module computes a feature vector composed of 34 foreground features. This feature vector is mapped to one of the 256 possible discrete symbols available in a codebook previously constructed by using the K-means vector quantization process [17,18]. The output of the FFE module is a sequence of discrete symbols representing a given numeral string. The length of the sequence corresponds to the width (number of columns) of the string bounding box.

The set of features is extracted from the foreground pixels of the string image. Even knowing that background information may provide a strong recognition performance, its use is avoided since the objective is to jointly maximize segmentation and recognition performances of the implicit segmentation strategy proposed in the SCB stage. To calculate background features, we need to know *a priori* the boundaries of each digit inside the string. However, this is one of the objectives of the SCB stage.

The proposed foreground feature set is composed of local and global features extracted from each string column. The local features are based on transitions from background to foreground pixels, and vice versa. For each transition, the mean direction and corresponding variance are obtained by means of the statistic estimators defined in [29]. These estimators are more suitable for directional observations, since they are based on a circular scale. For instance, given the directional observations $\alpha_1 = 10^\circ$ and $\alpha_2 = 359^\circ$, they provide a mean direction ($\bar{\alpha}$) of 0° instead of 180° calculated by conventional estimators. Let $\alpha_1, \dots, \alpha_i, \dots, \alpha_N$ be a set of directional observations with distribution $F(\alpha_i)$ and size N . Figure 3 shows that α_i represents the angle between the unit vector \overline{OP}_i

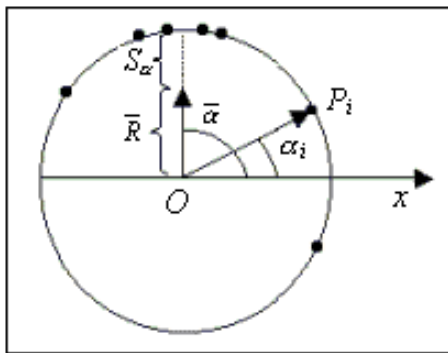


Fig. 3. Circular mean direction $\bar{\alpha}$ and variance S_α for a distribution $F(\alpha_i)$

and the horizontal axis, while P_i is the intersection point between $\overline{OP_i}$ and the unit circle.

The Cartesian coordinates of P_i are defined as:

$$(\cos(\alpha_i), \sin(\alpha_i)) \quad (1)$$

The circular mean direction $\bar{\alpha}$ of the N directional observations on the unit circle corresponds to the direction of the resulting vector (\bar{R}) obtained by the sum of the unit vectors ($\overline{OP_1}, \dots, \overline{OP_i}, \dots, \overline{OP_N}$). The center of gravity (\bar{C} , \bar{S}) of the N coordinates $(\cos(\alpha_i), \sin(\alpha_i))$ is defined as:

$$\bar{C} = \frac{1}{N} \sum_{i=1}^N (\cos(\alpha_i)) \quad (2)$$

$$\bar{S} = \frac{1}{N} \sum_{i=1}^N (\sin(\alpha_i)) \quad (3)$$

These coordinates are used to estimate the mean size of \bar{R} , as:

$$\bar{R} = \sqrt{(\bar{C}^2 + \bar{S}^2)} \quad (4)$$

Then, the circular mean direction can be obtained by solving one of the following equations:

$$\cos(\bar{\alpha}) = \frac{\bar{C}}{\bar{R}} \quad \text{or} \quad \sin(\bar{\alpha}) = \frac{\bar{S}}{\bar{R}} \quad (5)$$

Finally, the circular variance of $\bar{\alpha}$ is calculated as:

$$S_\alpha = 1 - \bar{R} \quad \text{with} \quad 0 \leq S_\alpha \leq 1 \quad (6)$$

To estimate $\bar{\alpha}$ and S_α for each transition of a numeral image, we have considered $\{0^\circ, 45^\circ, 90^\circ, 135^\circ, 180^\circ, 225^\circ, 270^\circ, 315^\circ\}$ as the set of directional observations, while $F(\alpha_i)$ is computed by counting the number of successive black pixels over direction α_i from a transition until the encounter of a white pixel. In Fig. 4 the transitions in a column of numeral 5 are enumerated from 1 to 6, and the possible directional observations from transitions 3 and 6 are shown. This directional information is very important since it give us some insight into the local shape of the digit stroke.

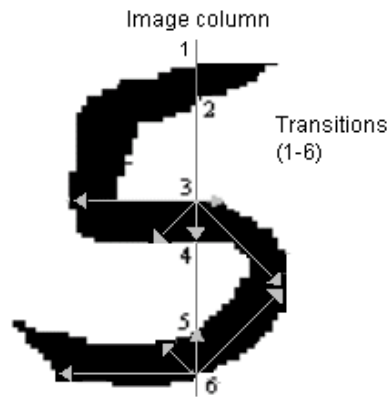


Fig. 4. Transitions in a column image of numeral 5, and the directional observations used to estimate the mean direction for transitions 3 and 6

In addition to this directional information, we have calculated two other local features: a) relative position of each transition with respect to the top of the digit bounding box, and b) whether the transition belongs to the outer or inner contour, which shows the presence of loops in the numeral image. The first feature is used to compensate for the lack of a zoning scheme in our feature extraction method. The second is used to detect the presence of loops, which are very discriminative for handwritten numerals. Since for each column eight possible transitions are considered, at this point the feature vector is composed of 32 features.

The global features are based on the vertical projection (VP) of black pixels for each column, and the derivative of VP between adjacent columns. These features provide the foreground pixel density of each column and the difference between adjacent columns that may give us some insight into the characteristic strokes of a digit. This constitutes a total of 34 features extracted from each column image and normalized between 0 and 1.

2.1.3 Segmentation-Recognition (SR) module. This module integrates segmentation and recognition through the use of an implicit segmentation-based strategy. It does so using the Level Building Algorithm (LBA) described in [6], which is responsible for matching numeral HMMs to the preprocessed string represented as a sequence of discrete symbols provided by the FFE module. In order to obtain the N -best candidate strings, we keep track of the K candidates at each level of the LBA instead of keeping just the globally best candidate as described in [6]. Since in the LBA, we have computed all reference pattern distances already (as needed to determine the best distance), all that is required is additional storage (to keep track of the array of K -best distances) and the book-keeping to determine the N -best strings, as described in details in [28].

To ensure an accurate representation of a numeral string, the numeral HMMs $(\lambda_c^0, \lambda_c^1, \dots, \lambda_c^9)$ are trained on a special data set extracted from the original NIST SD19. To create this data set, an automatic process based on a digit classifier is described in [2,3]. In this process, a hand-

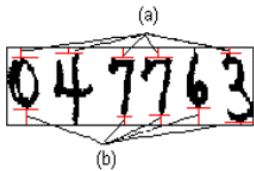


Fig. 5. Intra-string size variation: distances from top (a) and base (b) of the bounding box

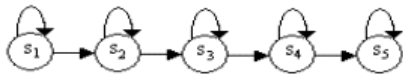


Fig. 6. Left-to-right HMM with 5 states

written numeral string is selected and segmented into digits when all of its components are recognized as isolated digits. Moreover, the string recognition result must correspond to that labeled by NIST. The objective is to obtain a data set in which the isolated digits have a link to their original strings. This enables the use of string contextual information during training of numeral models on the isolated digits. The general idea is to keep the same experimental conditions during system training and testing.

Contextual information used during training concerns string slant and intra-string size variations. To consider slant contextual information, we use the slant estimated from the original string to correct the isolated digits used for training the implicit segmentation-based method used in the SCB stage.

To deal with intra-string size variation, we have considered the distances or blank spaces on top and bottom of some digits in the string (see Fig. 5) as contextual information. For this purpose, features are extracted from each training sample (isolated digit) by taking into account the height of its original string bounding box instead of the height of its own bounding box. Moreover, to deal with inter-string size variation, we use only size-invariant features in the FFE module.

The use of contextual information has shown to be a promising strategy to obtain an accurate representation of numeral strings by using models based on isolated digits. We can see the contribution to recognition performance in Sect. 3.

2.1.4 Numeral models. The topology of the numeral models is defined taking into account the recognition of handwritten text and the use of LBA. This means a left-right model without initial or end-states. Figure 6 shows the initial 5-state HMM topology used in the baseline SCB stage, in which the number of states was experimentally defined.

The same, or similar, topology can be found in related works. In [6], the authors use it to model character classes to recognize fax-printed words. The same structure is used for modeling numeral classes to recognize handwritten numeral strings in [26]. Another similar HMM topology, with additional skip transitions, is used for modeling airline vocabulary in [28]. In all the above, the LBA is used as a recognition algorithm.

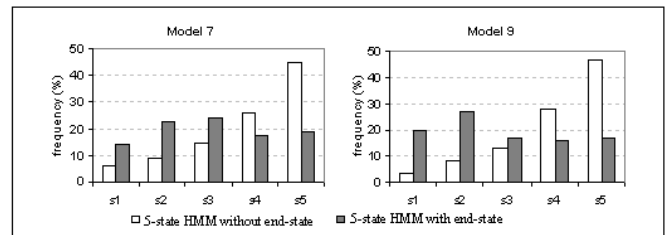


Fig. 7. Distributions of observations among the HMM states computed during model training

As we can see, the HMM topology used in the baseline system does not present additional states (initial or end-states) to enable the concatenation of numeral models, since these are not necessary in the LBA framework. In the SR module, 10 numeral models independently trained on isolated numerals are used to recognize strings, and the LBA is responsible for finding the best sequence of these models for a given numeral string. However, this topology does not allow us to model the interaction between adjacent numerals in strings. Moreover, in the preliminary experiments on numeral strings in Sect. 3, we can observe a significant loss in terms of recognition performance as the string length increases (see Table 5). In order to better understand the behavior of these numeral models, the distribution of observations among the HMM states is computed during their training on 50,000 isolated numerals (5,000 samples per class).

Figure 7 presents the distributions for digit classes seven and nine as a 5-state HMM without end-state. These unbalanced distributions of observations among the states, associated with the presence of a self-transition with probability value equal to 1.0 in the last HMM state (s5), have a negative impact on the segmentation performance of the SR module, as described in [3]. To deal with this problem and also adapt the numeral models to a string-based training, we include an end-state in the HMM topology (see Fig. 8). The new models show a better distribution of the observations among their states, as we can see in Fig. 7 (5-state HMM with end-state), and avoid a self-transition with probability value equal to 1.0 in the state 5 (s5). The end-state does not absorb any observation, and it is useful to concatenate the numeral models during a string-based training. The positive impact of this modification on the HMM topology to the string recognition is shown in Sect. 3 and also in [3].

Based on this new topology, we can pay some attention to the possibility of integrating handwriting-specific knowledge into the model structure to obtain an accurate representation of numeral strings. In addition, the final length of each numeral HMM (number of states) was re-defined through the use of the scheme described in [32]. This scheme estimates a range for the length of each numeral HMM taking into account statistics estimated from the length of the observation sequences on the training database. Table 1 shows the final length of each numeral model.

2.1.5 Space model. With the objective of obtaining an accurate representation of numeral strings, we investigate

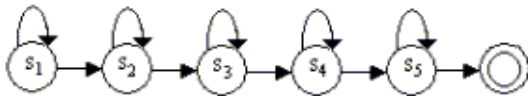


Fig. 8. 5-state HMM with an end-state

Table 1. Final length of each numeral HMM

| Numeral model | Number of states |
|---------------|------------------|
| 0 | 13 |
| 1 | 6 |
| 2 | 14 |
| 3 | 14 |
| 4 | 15 |
| 5 | 13 |
| 6 | 15 |
| 7 | 15 |
| 8 | 14 |
| 9 | 16 |

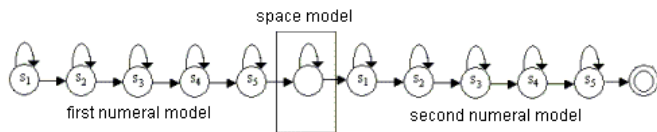


Fig. 9. Concatenation of numeral models during string-based training

a way of integrating some contextual information in the numeral models regarding the inter-digit spaces. For this purpose, we use a two-step training mechanism, in which numeral models previously trained on isolated digits are submitted to a string-based training. In the second step of this training mechanism, a space model is built into the numeral models.

The strategy of building the space model into the numeral models instead of using an independent model is an important one in the LBA framework, since possible future problems when the method is generalized for unknown-length strings will be avoided. An independent space model would represent one more model competing at each LBA level, which must be taken into account to estimate the string length (number of digits).

The space model is trained on digit pairs extracted from the NIST database. In this training, for a given digit pair the corresponding numeral models are concatenated by using the end-state. In fact, the end-state of the first model is replaced with the first state of the second model (see Fig. 9). For the space model experiment, we use a two step-training mechanism: 1) 10 numeral models are trained first on isolated digits, 2) the numeral models are submitted to a string-based training using digit pairs (DPs) extracted from the NIST database.

The DP database is balanced in terms of number of naturally segmented, overlapping and touching numerals. The NIST series *hsf.0* to *hsf.3* were used to provide the training and validation samples. Table 2 shows the number of samples in the training and validation sets, and also presents the number of samples representing each string class.

Just the space model parameters are estimated during the second-step training. The parameters corresponding

Table 2. Digit pair database (strings composed of 2 digits)

| String samples | Training | Validation |
|---------------------|----------------|---------------|
| Naturally segmented | 8,000(53.3%) | 1,800(51.4%) |
| Touching digits | 4,000(26.7%) | 1,000(28.6%) |
| Overlapping digits | 3,000(20.0%) | 700(20.0%) |
| Total | 15,000(100.0%) | 3,500(100.0%) |

to the numeral models are kept the same as estimated during the first training step based on isolated numerals. The corresponding experimental results are reported in Sect. 3.

2.2 Verification stage

The second stage of the proposed method consists of two modules: the Foreground/Background Feature Extraction (FBFE) and the Verification modules. The main component of this stage consists of an HMM-based digit classifier trained on isolated digits without taking into account any string contextual information. A new set of features combines foreground and background information to provide numeral HMMs with high recognition performance. Moreover, 10 additional numeral HMMs ($\lambda_{vr}^0, \lambda_{vr}^1, \dots, \lambda_{vr}^9$) based on the rows of the numeral images are combined with the column-based models ($\lambda_{vc}^0, \lambda_{vc}^1, \dots, \lambda_{vc}^9$) to accurately represent the digit classes. The objective is to use these new models and the Viterbi's algorithm [28] to verify and re-rank the segmentation-recognition paths provided by the SCB stage.

2.2.1 Foreground/Background Feature Extraction (FBFE) module. The verification stage starts in this module, in which the segmentation points provided by the first stage are used to define string segments and calculate the corresponding bounding boxes. Then, for a given segment, this module extracts a feature vector combining foreground and background information for each column in the segment bounding box. This feature vector is mapped to one of 256 possible discrete symbols available in a codebook previously constructed by using the K-means algorithm. A similar process is carried out for the rows of the segment. Thus, the output of the FBFE module consists of two sequences of discrete observations for each segment: column-based and row-based sequences.

The feature vector computed for each column and for each row of the digit image is composed of 47 features: the 34 foreground features used in the SR module, plus 13 background features. The background features are based on concavity information. These features are used to highlight the topological and geometrical properties of the digit classes. Each concavity feature represents the number of white pixels that belong to a specific concavity configuration.

The label for each white pixel is chosen based on the Freeman code with four directions. Each direction is explored until the encounter of a black pixel or the limits imposed by the digit bounding box. A white pixel is

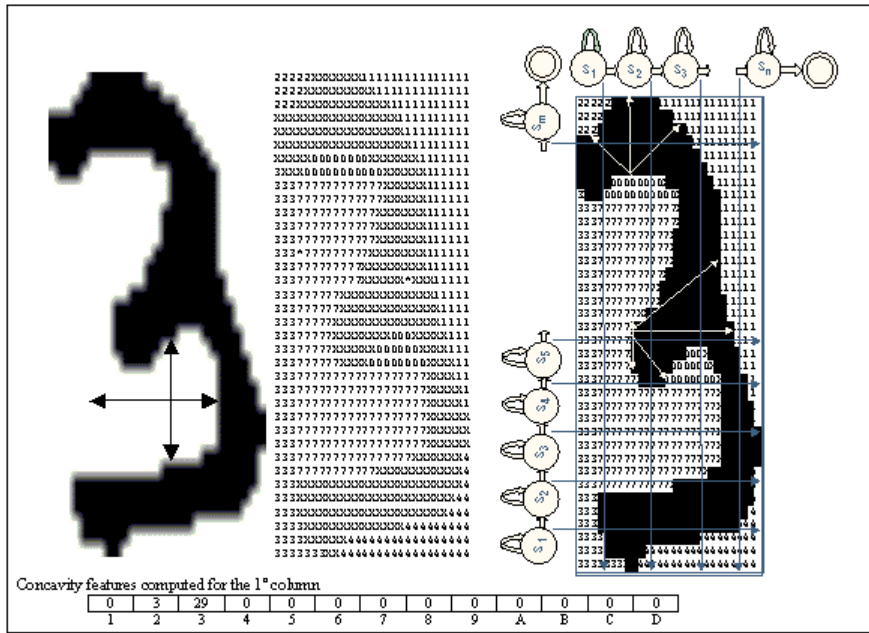


Fig. 10. Example of concavity features and the zoning scheme provided by column and row models

labeled if at least two consecutive directions find black pixels. Thus, we have 9 possible concavity configurations (see Fig. 10). Moreover, we consider four more configurations, in order to detect more precisely the presence of loops. The total length of this feature vector is then 13. The concavity vector is normalized between 0 and 1, by the total of the concavity codes computed for each column or row of the digit image. The column-based and row-based models provide a way of combining foreground and background features in the zoning scheme as shown in Fig. 10.

2.2.2 Verification module. This module is based on the Viterbi’s algorithm described in [28]. Each digit class is represented by two numeral HMMs: one based on the image columns ($\lambda_{vc}^0, \lambda_{vc}^1, \dots, \lambda_{vc}^9$) and other based on the image rows ($\lambda_{vr}^0, \lambda_{vr}^1, \dots, \lambda_{vr}^9$) of digit images. In this module, the first step concerns the selection of the column and row models corresponding to the string segment to be verified. This is done using the segment recognition result provided by the SCB stage. The selected pair of models is scored using the Viterbi’s algorithm and the sequence of observations extracted by the FBFE module. The output probabilities of both the column and row models are combined by summing their logs. This results in a segment score. A string score can be obtained by summing the score of each segment. This new string score calculated in the Verification module is added to the score of the segmentation-recognition hypothesis obtained from the SCB stage. The final resulting score is used to re-rank the string segmentation-recognition hypothesis.

To explain this further, let us to consider the N segmentation-recognition hypotheses provided by the SCB stage as (srh_1, \dots, srh_N) , where each srh_i is a structure composed of the following fields: string length (M), segmentation points $(sp_{i1}, \dots, sp_{iM-1})$, recognition result of each string segment (r_{i1}, \dots, r_{iM}) , and the score

of the segmentation-recognition hypothesis ($SCBsrh_i$). In the Verification stage, the first step carried out by the FBFE module, in which the segmentation points $(sp_{i1}, \dots, sp_{iM-1})$ are used to define each segment (seg_{ij}) and its bounding box. A feature vector is extracted from each column and row of seg_{ij} taking into account its bounding box. The column and row based feature vectors are quantized in order to generate two sequences of discrete observations representing each seg_{ij} , respectively (see Fig. 11). The Verification module uses the recognition result r_{ij} of the corresponding segment to select the column- and row-based numeral HMMs, $\lambda_{vc}^{r_{ij}}$ and $\lambda_{vr}^{r_{ij}}$, which are scored using the respective sequence of observations. A segment recognition score ($segsc_{ij}$) is estimated by summing the log of the probability of the column and row numeral models. The verification string score is obtained by:

$$Vstr_i = \sum_{j=1}^M segsc_{ij} \tag{7}$$

Finally, the segmentation-recognition score of the SCB stage ($SCBsrh_i$) and that obtained in the Verification stage ($Vstr_i$) are combined as:

$$Finalstr_i = SCBsrh_i + Vstr_i \tag{8}$$

The resulting string score ($Finalstr_i$) is used to re-rank the i^{th} string recognition hypothesis.

2.2.3 Numeral models. The digit classes are represented by 20 models: 10 column-based ($\lambda_{vc}^0, \lambda_{vc}^1, \dots, \lambda_{vc}^9$) and 10 row-based ($\lambda_{vr}^0, \lambda_{vr}^1, \dots, \lambda_{vr}^9$) models. They have the same end-state topology used for the numeral models of the SR module. The optimization scheme defined in [32] is also used to define the final length (number of states) of these models (see Table 3). However, they differ from

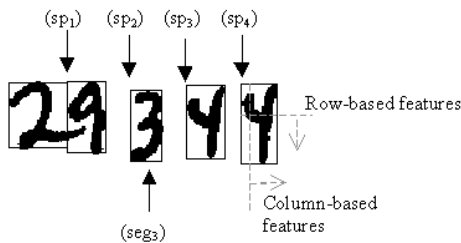


Fig. 11. A numeral string, the segmentation points (sp_i), a segment and its corresponding bounding box (seg_3) and the column and row-based feature extraction

Table 3. Number of states of the numeral models

| Digit class | Column-based model | Row-based model |
|-------------|--------------------|-----------------|
| 0 | 13 | 14 |
| 1 | 6 | 16 |
| 2 | 14 | 16 |
| 3 | 14 | 20 |
| 4 | 15 | 18 |
| 5 | 13 | 19 |
| 6 | 15 | 18 |
| 7 | 15 | 18 |
| 8 | 14 | 20 |
| 9 | 16 | 21 |

the SR models in the sense that they are trained without considering any string contextual information and using a feature set based on both foreground and background information.

Moreover, there is no space model inside them. The objective is to obtain numeral models which are more powerful in terms of isolated digit recognition performance than those used in the SR module.

We can see in both stages of the proposed method the use of discrete HMMs. We have decided to implement discrete HMMs to be sure that in both SCB and Verification stages, there will be enough data for training. This strategy is important to allow the modelling of specific handwriting-knowledge related to the interaction between adjacent digits in strings (such as inter-digit spaces). In addition, it is also important to ensure portability to the proposed method.

3 Experimental results

In this section, experiments undertaken during the course of development of the proposed method are detailed. In the first set of experiments, string recognition is based on an informed strategy, i.e., the string length (number of digits) is known. The objective of using this strategy is to evaluate the system under different conditions, while at the same time adjusting some important aspects regarding string normalization, feature extraction and HMM parameters. A non-informed strategy is used in the final experiments. It is important to point out that all the experiments were conducted considering a zero-level rejection.

The protocol used to implement and evaluate the proposed method consists of three steps. In the first

step, which is called SCB Stage Construction and Evaluation, we construct a baseline system composed of the Preprocessing, Forward Feature Extraction (FFE) and Segmentation-Recognition (SR) modules representing the first stage of the proposed numeral string recognition method. The second step of the evaluation protocol is called Verification Stage Construction and Evaluation. The objective is to re-rank the N best segmentation-recognition hypotheses of the SCB stage by using a verification strategy. The general idea here being to evaluate the SCB stage with respect to segmentation and recognition of numeral strings, and to include a further verification step to check and re-rank its results. In the last step of the evaluation protocol the system is evaluated using a non-informed strategy, where the string length is unknown. An error analysis is also presented.

3.1 Databases

The isolated numerals used in these experiments come from the NIST SD19, we use 50,000 numeral samples for training, 10,000 for validation and 10,000 for testing. The training samples were extracted from *hsf_0*, *hsf_1* and *hsf_2*, the validation samples from *hsf_7* and the testing samples from *hsf_4*. The codebooks used in the proposed method were generated based on the 50,000 numeral samples of the training set (5,000 per class). In the SCB stage, for example, since a feature vector is calculated for each image column, more than 1,500,000 feature vectors were used in the vector quantization process (considering around 30 columns per image in average). A similar number of feature vectors is processed during the construction of the codebooks used in the Verification Stage (column and row-based codebooks).

The experiments using numeral strings are based on 12,802 samples extracted from the *hsf_7* series of NIST SD19 and distributed into 6 classes of strings: 2_digit (2,370), 3_digit (2,385), 4_digit (2,345), 5_digit (2,316), 6_digit (2,169) and 10_digit (1,217). These strings exhibit different problems, such as touching, overlapping and fragmentation. In addition, to evaluate the system in terms of touching digits we use a subset of data containing 2,069 touching digit pairs (TDPs) also extracted from NIST SD19.

3.2 SCB stage – construction and evaluation

This section corresponds to the first step of the evaluation protocol, in which the SCB stage is improved at each new experiment. In the experiments on string normalization, both slant normalization techniques with and without contextual information are developed and evaluated. Moreover, the advantage of using contextual information instead of a size normalization method to deal with the intra-string size variation is shown. These experiments also show that the foreground features proposed in the FFE module are really unaffected by inter-string size variation.

Table 4. Slant normalization experiments – isolated digit recognition(%)

| Experiments | Validation | Testing |
|--------------------------------|------------|---------|
| Without slant normalization | 93.1 | 86.2 |
| Slant normalization without CI | 95.9 | 93.0 |
| Slant normalization with CI | 95.8 | 92.6 |

Table 5. Slant normalization experiments – string recognition results (%)

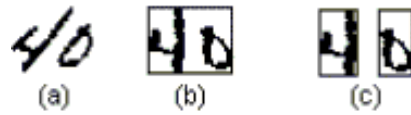
| String class | Without slant normalization | Slant normal. without CI | Slant normal. with CI |
|--------------|-----------------------------|--------------------------|-----------------------|
| 2_digit | 77.6 | 80.9 | 84.2 |
| 3_digit | 66.3 | 70.6 | 75.1 |
| 4_digit | 57.0 | 64.4 | 69.2 |
| 5_digit | 49.3 | 53.1 | 61.1 |
| 6_digit | 43.5 | 47.3 | 58.1 |
| 10_digit | 17.4 | 31.0 | 34.4 |
| Global | 55.1 | 60.5 | 66.4 |

In addition, we evaluate the impact of adding an end-state to the HMM structure. Some experiments show the improvement obtained by considering a space model built into the numeral models. In the final experiments, the HMM parameters are optimized.

3.2.1 Baseline system. The experiments start with the construction of a baseline system, which corresponds to the SCB stage. In this version, the HMM parameters and codebook size were experimentally defined. The best results were obtained with *5-states* discrete left-right numeral HMMs and a codebook of 64 entries. The feature vector is composed of foreground features extracted from the image columns (34-vector) as previously described. The SR module corresponds to an LBA. It is used to give the best segmentation-recognition path for a given numeral string. There is no verification module in the baseline system.

3.2.2 Experiments on slant normalization. These experiments are designed to answer the following question: what is the real contribution to numeral string recognition achieved by using string slant to normalize isolated digits used to train the numeral models of the SCB stage? To this end, the baseline system is used to compare recognition performance by considering no slant normalization, slant normalization without contextual information and with contextual information (CI). In the experiment based on slant normalization without contextual information, each isolated numeral used for training the numeral HMMs is slant-corrected using the slope estimated from its own image. In contrast, when contextual information is used, the slope for each isolated numeral is estimated as the slope calculated for its original string.

Table 4 presents some preliminary recognition results for isolated digits. We can observe that contextual information does not contribute to the recognition of slant normalized isolated numerals, since the digit origin is not helpful information in this case.

**Fig. 12.** a Original string; b String bounding box after slant normalization; c Training samples linked to their original string and the bounding box used for feature extraction**Table 6.** Size normalization experiments – isolated digit recognition(%)

| Experiments | Validation | Testing |
|----------------------------|------------|---------|
| Without size normalization | 95.8 | 92.6 |
| Size normalization | 95.2 | 92.2 |
| IntraSSV | 94.8 | 91.1 |

Table 7. Size normalization experiments – string recognition results (%)

| String class | Without size normalization | Size normalization | IntraSSV |
|--------------|----------------------------|--------------------|----------|
| 2_digit | 84.2 | 84.2 | 85.3 |
| 3_digit | 75.1 | 75.6 | 78.1 |
| 4_digit | 69.2 | 69.9 | 71.3 |
| 5_digit | 61.1 | 64.0 | 66.3 |
| 6_digit | 58.1 | 59.7 | 63.8 |
| 10_digit | 34.4 | 40.7 | 44.0 |
| Global | 66.4 | 68.0 | 70.4 |

On the other hand, Table 5 shows very interesting results for numeral strings. Normalizing without contextual information has brought an improvement of 5.4% in the global string recognition rate, while the use of contextual information allows an improvement of 11.3%, both compared to the experiments without slant normalization

3.2.3 Experiments on size normalization. In these experiments, we evaluate two strategies to deal with the intra-string size variation: a) the use of a non-linear size normalization method described in [16], to normalize each numeral string along the vertical axis by using the mean digit height (45 pixels) calculated from the training database; b) the use of contextual information regarding intra-string size variation (IntraSSV) during training of the numeral models. In the last strategy, features from each training sample representing an isolated digit are extracted taking into account the height of its original string bounding box instead of the height of its own bounding box (see Fig. 12). Moreover, we consider the foreground features as size invariant. This means that nothing is done to deal with inter-string size variation.

Table 6 shows that the foreground features really do not vary with inter-string size variation. Moreover, the experiment based on IntraSSV loses in terms of numeral recognition, which is possible since the use of this additional contextual information increases the numeral variability. In contrast, this experiment brought a further improvement of 4% to the global string recognition rate (see Table 7), even with a small loss in terms of isolated numeral recognition performance.

The experiments considering the use of a size normalization method have shown some improvement, but

Table 8. End-state experiments – string recognition results (%)

| String class | HMM without end-state | HMM with end-state |
|--------------|-----------------------|--------------------|
| 2_digit | 85.3 | 87.7 |
| 3_digit | 78.1 | 82.4 |
| 4_digit | 71.3 | 78.1 |
| 5_digit | 66.3 | 75.6 |
| 6_digit | 63.8 | 71.6 |
| 10_digit | 44.0 | 60.6 |
| Global | 70.4 | 77.5 |

not so relevant as the one brought by the use of IntraSSV for training the numeral models. This is due to additional touching digits and distortions in the digit strokes caused by the size normalization method.

3.2.4 Contribution of an end-state in the HMM topology.

These experiments show that the HMM topology with end-state does not bring a significant improvement in the recognition of isolated numerals (about 0.7%). On the other hand, it brought about a 7.1% improvement in the global string recognition rate (see Table 8).

This is due a better distribution of the observations among the HMM states, and a better estimation of the self-transition probability in the last HMM state. Consequently, the LBA provides a more precise match of numeral models to the observation sequence. This means a better definition of string segmentation points.

3.2.5 Contribution of a space model. In these experiments, the digit pair database and the two-step training mechanism described in Sect. 2 are used to evaluate the use of a space model. This corresponds to an additional state in the numeral HMM structure. The following strategies are evaluated: 1) the use of one space model for each numeral class; and 2) the use of one space model representing all numeral classes. In both strategies, the numeral models are first trained on isolated digits. Subsequently, the space model parameters are estimated during the second-step training on digit pairs. The parameters corresponding to the numeral models are kept the same as estimated during the first training step on isolated numerals.

Table 9 summarizes all these experiments. It can be observed that the space model brings about some improvement for each string class. The recognition performance of both strategies is almost the same, which shows that the space model is not dependent on digit class.

3.2.6 Optimization of the HMM parameters. The scheme described in [32] for defining the HMM length is used to redefine the number of states of the numeral HMMs in the SCB stage. In addition, a new codebook is evaluated. The best results are achieved by using the values shown in Table 3, and a codebook with 256 entries. Table 10 shows the impact on the recognition performance for isolated digits, while Table 11 does so for numeral strings. In these experiments, we did not consider the space model.

Table 9. Space model experiments – string recognition results (%)

| String class | HMM with end-state | HMM with end-state and space model* | HMM with end-state and space model** |
|--------------|--------------------|-------------------------------------|--------------------------------------|
| 2_digit | 87.7 | 87.9 | 87.9 |
| 3_digit | 82.4 | 82.7 | 82.6 |
| 4_digit | 78.1 | 78.4 | 78.4 |
| 5_digit | 75.6 | 76.0 | 76.1 |
| 6_digit | 71.6 | 72.0 | 72.0 |
| 10_digit | 60.6 | 61.1 | 61.3 |
| Global | 77.5 | 77.8 | 77.8 |

(*one space model by digit class)

(**one space model for all digit classes)

Table 10. Different HMM configurations – isolated digit recognition(%)

| Experiments | Validation | Testing |
|-------------------------------|------------|---------|
| Baseline system | 95.6 | 91.7 |
| Optimized HMMs & 256-codebook | 96.7 | 94.0 |

Table 11. HMM parameters optimization – string recognition results (%)

| String class | Baseline system | Optimized system |
|--------------|-----------------|------------------|
| 2_digit | 87.7 | 89.8 |
| 3_digit | 82.4 | 84.4 |
| 4_digit | 78.1 | 80.3 |
| 5_digit | 75.6 | 78.0 |
| 6_digit | 71.6 | 75.8 |
| 10_digit | 60.6 | 66.2 |
| Global | 77.5 | 80.3 |

We update the space model experiments considering the new HMM parameters. One space model representing all numeral classes is considered in Table 12.

3.3 Verification stage – construction and evaluation

As previously indicated, the verification module is composed of 20 numeral HMMs: 10 based on the columns and 10 based on the rows of the digit images. These complementary HMM models are used as an isolated digit classifier for re-ranking the segmentation-recognition hypotheses provided by the SCB stage.

The same scheme used for optimizing the numeral HMMs of the SR module is applied to define the length of these new HMM models. The best result is obtained by using the values in Table 3. The codebook composed of 256 entries has provided the best results. Table 13 shows the recognition results for isolated digits when the column and row models are combined. They are combined by summing the log of the final probability of each model calculated using Viterbi’s algorithm.

Table 12. String recognition results after optimizing the HMM parameters and using a space model (%)

| String class | System without space model | System with space model |
|--------------|----------------------------|-------------------------|
| 2_digit | 89.8 | 90.2 |
| 3_digit | 84.4 | 85.8 |
| 4_digit | 80.3 | 81.6 |
| 5_digit | 78.0 | 79.9 |
| 6_digit | 75.8 | 76.7 |
| 10_digit | 66.2 | 68.4 |
| Global | 80.3 | 81.6 |

Table 13. Combination of column and row models – isolated digit recognition(%)

| | Validation | Testing |
|---------------------------|------------|---------|
| Column based features | 98.4 | 96.5 |
| Row based models | 98.4 | 97.0 |
| Combination(column x row) | 99.0 | 98.0 |

Table 14. SCB stage – numeral string recognition results (%)

| Class | Top (1) | Top (2) | Top (3) | Top (4) | Top (5) |
|----------|---------|---------|---------|---------|---------|
| 2_digit | 90.2 | 95.3 | 96.9 | 97.2 | 97.4 |
| 3_digit | 85.8 | 91.9 | 92.8 | 93.2 | 93.3 |
| 4_digit | 81.6 | 89.3 | 91.1 | 91.8 | 91.9 |
| 5_digit | 79.9 | 87.6 | 89.5 | 90.5 | 90.6 |
| 6_digit | 76.7 | 85.8 | 87.3 | 88.4 | 88.8 |
| 10_digit | 68.4 | 73.6 | 74.2 | 74.4 | 74.4 |
| Global | 81.6 | 88.5 | 90.0 | 90.6 | 90.8 |
| TDPs | 79.5 | 88.4 | 91.6 | 92.6 | 93.1 |

3.4 Recognition results of known-length strings

During these experiments the SR module provides the 10 best segmentation-recognition paths for each numeral string. In the Verification stage, the FBF module uses the segmentation points of each path as delimiters in the preprocessed string image to calculate new features based on columns and rows for each string segment. The recognition result of the first stage is verified using the new set of features and numeral HMMs available in the Verification stage. We combine the recognition results of the SCB and Verification stages as described in Sect. 2.2. Table 14 shows the top 5 recognition results of the first stage of our system, while Table 15 presents the top 5 recognition results after the Verification stage. The last line of these tables shows the recognition results for Touching Digit Pairs (TDPs) using a database composed of 2,069 samples extracted from NIST database.

We can see a significant improvement in the recognition performance by using the Verification stage. The main reason is that the foreground features and the numeral HMMs based on contextual information of the SCB stage may contemplate both segmentation and recognition tasks in an implicit segmentation approach, but they do not provide a strong enough recognition power.

Table 15. SCB + Verification stage – numeral string recognition results (%)

| Class | Top (1) | Top (2) | Top (3) | Top (4) | Top (5) |
|----------|---------|---------|---------|---------|---------|
| 2_digit | 95.2 | 97.5 | 98.3 | 98.4 | 98.5 |
| 3_digit | 92.6 | 95.6 | 96.1 | 96.2 | 96.2 |
| 4_digit | 92.1 | 95.3 | 95.9 | 96.0 | 96.1 |
| 5_digit | 90.0 | 93.9 | 94.5 | 94.6 | 94.7 |
| 6_digit | 90.0 | 94.0 | 94.8 | 94.9 | 95.0 |
| 10_digit | 86.9 | 90.3 | 90.3 | 90.4 | 90.4 |
| Global | 91.5 | 94.8 | 95.4 | 95.5 | 95.6 |
| TDPs | 89.6 | 94.3 | 95.3 | 95.7 | 95.8 |

3.5 Recognition results of unknown-length strings

So far, the string recognition has been based on an informed strategy, i.e., the string length (number of digits) is known. The objective of using this strategy was to evaluate the system in different conditions, while at the same time adjusting some important aspects regarding string normalization, feature extraction and HMM parameters. In the experiments reported in this section, a non-informed strategy is used, i.e., the string length is unknown.

To deal with this problem, we have defined a string length predictor based on Bayes theory described in [11]. It uses the minimum-error-rate decision rule to predict the string length (number of digits) given the width of the string bounding box (sbb) in terms of number of columns. A set of string classes is defined as

$$w = \{2_digit, 3_digit, 4_digit, 5_digit, 6_digit, 10_digit\},$$

in which class $\#_digit$ corresponds to strings composed of $\#$ digits. The a priori probabilities of these classes are considered ambiguous, i.e., $P(2_digit) = P(3_digit) = P(4_digit) = P(5_digit) = P(6_digit) = P(10_digit)$. The parameters of a Gaussian *pdf* are estimated for each class by using a training set composed of 44,256 handwritten numeral strings extracted from the NIST SD19 database. Then, a string length classifier is designed to classify the *sbb_width* into M classes of string lengths by using M discriminant functions $g_i(sbb_width)$, computing the similarities between the unknown data *sbb_width* and each string class w_j , and selecting the class w_i corresponding to

$$g_i(sbb_width) > g_j(sbb_width) \quad \text{for all } j \neq i \quad (9)$$

Figure 13 shows the scheme of this classifier, where the decision rule is to maximize the a *posteriori* probability.

The same testing set composed of 12,802 numeral strings used to evaluate the recognition method is used to test this string length predictor. Figure 14 shows the classification results. It is possible to observe that when the right decision is considered in the best 3 hypotheses, the performance of the string length predictor is very promising. Thus, the top 3 decisions of the string

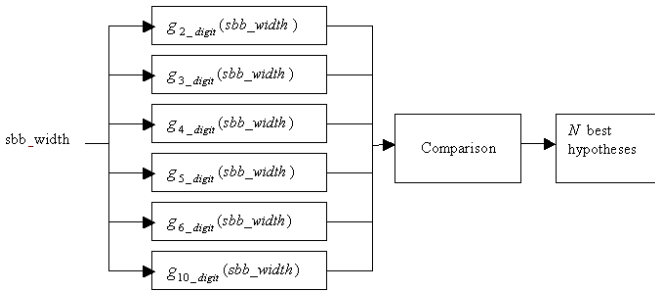


Fig. 13. The classifier used to predict the string length from the width of the string bounding box (*sbb_width*)

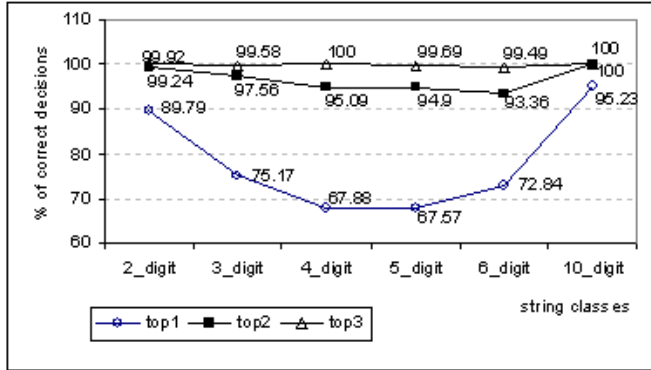


Fig. 14. Classification results of the *sbb_width* into length classes using the proposed classifier

Table 16. Recognition of unknown-length strings using the length predictor (%)

| Class | Top (1) | Top (2) | Top (3) | Top (4) | Top (5) |
|----------|---------|---------|---------|---------|---------|
| 2_digit | 94.8 | 97.1 | 97.9 | 98.0 | 98.1 |
| 3_digit | 91.6 | 94.6 | 95.0 | 95.0 | 95.0 |
| 4_digit | 91.2 | 94.2 | 94.8 | 94.9 | 94.9 |
| 5_digit | 88.3 | 92.1 | 92.6 | 92.7 | 92.8 |
| 6_digit | 89.0 | 92.8 | 93.5 | 93.5 | 93.6 |
| 10_digit | 86.9 | 90.3 | 90.3 | 90.4 | 90.4 |
| Global | 90.6 | 93.8 | 94.4 | 94.5 | 94.5 |
| TDPs | 88.9 | 93.5 | 94.8 | 95.3 | 95.7 |

length predictor are used to determine the L parameter of the LBA. For instance, if the top 3 decisions belong to the *3_digit*, *4_digit* and *5_digit* classes, then five levels are constructed by the LBA and after that the probabilities of each segmentation-recognition hypothesis corresponding to these string lengths are compared. The segmentation-recognition hypothesis with the highest probability is chosen. Table 16 shows the recognition results when the string length is predicted using the proposed string length classifier.

3.6 Error analysis

Table 17 shows the confusion matrix computed from the isolated digit recognition results of the SCB stage. This matrix confirms that the numeral models of the SCB stage are not powerful enough in terms of recognition

Table 17. Confusion matrix – isolated digit recognition of the SCB stage

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | 929 | 0 | 1 | 0 | 9 | 0 | 20 | 0 | 5 | 8 |
| 1 | 0 | 980 | 1 | 1 | 0 | 0 | 0 | 2 | 0 | 1 |
| 2 | 9 | 9 | 969 | 4 | 3 | 1 | 1 | 74 | 5 | 7 |
| 3 | 0 | 0 | 21 | 980 | 0 | 75 | 3 | 16 | 2 | 4 |
| 4 | 6 | 5 | 1 | 3 | 950 | 2 | 8 | 13 | 0 | 30 |
| 5 | 7 | 1 | 0 | 7 | 0 | 897 | 26 | 0 | 2 | 2 |
| 6 | 29 | 0 | 2 | 0 | 6 | 0 | 909 | 0 | 1 | 0 |
| 7 | 0 | 4 | 1 | 3 | 13 | 0 | 0 | 874 | 0 | 2 |
| 8 | 4 | 0 | 4 | 2 | 2 | 18 | 33 | 12 | 981 | 15 |
| 9 | 16 | 1 | 0 | 0 | 17 | 7 | 0 | 9 | 4 | 931 |

Table 18. Confusion matrix – isolated digit recognition of the Verification stage

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | 988 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 2 | 0 |
| 1 | 0 | 986 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 2 | 1 | 8 | 993 | 3 | 0 | 0 | 0 | 21 | 0 | 0 |
| 3 | 1 | 0 | 1 | 995 | 0 | 7 | 0 | 12 | 1 | 2 |
| 4 | 2 | 2 | 1 | 0 | 983 | 0 | 0 | 2 | 0 | 23 |
| 5 | 0 | 1 | 0 | 1 | 0 | 971 | 24 | 1 | 0 | 1 |
| 6 | 6 | 1 | 0 | 0 | 2 | 0 | 966 | 0 | 1 | 0 |
| 7 | 0 | 1 | 4 | 0 | 1 | 0 | 0 | 961 | 0 | 1 |
| 8 | 2 | 1 | 1 | 1 | 0 | 12 | 6 | 2 | 995 | 9 |
| 9 | 0 | 0 | 0 | 0 | 14 | 10 | 0 | 1 | 1 | 964 |

performance of isolated digits. Their weakness are related to the feature extraction method used in this stage, the objective of which is to maximize the likelihood of segmentation and recognition of numeral strings in an implicit segmentation-based process. Lots of confusions are shown, e.g., 2-7, 3-5, and many others in Table 17. However, we observed that the SCB stage is often able to find the right segmentation points for a given string, even without achieving the right recognition. This is possible, given the similar length of the observation sequences representing the digit classes involved in these confusing situations.

In order to overcome this problem, more discriminative features are necessary, such as holes, concavities or zoning-based features. However, most of these features require the digit bounding box or the boundaries of each digit inside the string to be calculated. Otherwise, one digit may interfere with the calculation of the adjacent ones. However, finding the digit boundaries in the string is the objective of the SCB stage.

The confusion matrix in Table 18 shows that combining column and row numeral models to represent each digit class provides an interesting recognition performance. The background features based on concavities have shown to be a promising way to distinguish between classes 2-7 and 3-5. However, there are still some confusions between classes, e.g., 2-7, 4-9, and 5-6.

Table 19 shows the system mistakes related to the recognition of handwritten numeral strings categorized as: a) segmentation caused by touching problems; b) segmentation caused by overlapping problems; c) digit recognition; and d) to the presence of noise in the string im-

Table 19. Summary of the system mistakes (%)

| | | | |
|--------------|-------------------|------|------|
| Segmentation | Touching | 21.2 | |
| | Overlapping | 11.6 | |
| | Natural segmented | 0.8 | 33.6 |
| Recognition | | | 62.1 |
| Noise | | | 4.3 |

Table 20. Examples of incorrectly recognized strings (some examples of segmentation cuts)































| | | |
|---|---|--|
|  84187 (84297) |  90398(90898) |  03223(03283) |
|  32072 (37077) |  88 (86) |  95537(45537) |
|  47534 (97539) |  13211 (13210) |  60114(60118) |
|  77478 (72478) |  72937(12931) |  78434(75434) |

Table 21. Examples of correctly recognized strings

| | | |
|---|---|---|
|  54593 |  8419 |  22699 |
|  2024 |  3258 |  65381 |
|  7433 |  50 |  69878 |
|  25 |  25 |  20356 |
|  56 |  76540 |  43733 |
|  92174 |  20044 |  48890 |

ages. The segmentation mistakes are most of the time related to touching or overlapping problems.

As expected the worst segmentation problem concerns touching digits (21.2%). Few segmentation mistakes (0.8%) occurred in naturally segmented strings and they are due to the lack of samples for training the space model. The digit pair database is not representative of strings of different lengths.

However, most of the mistakes in the applying the method are due to recognition problems (62.1%). Few mistakes are related to presence of noise in the string images. Table 20 shows some examples of incorrectly recognized strings, while Table 21 presents some examples of correctly recognized strings.

4 Discussion

The first set of experiments have shown that the use of contextual information to provide the same conditions during training and testing is a promising strategy in an implicit segmentation-based method. The use of slant normalization brought about an improvement of 5.4% to the global string recognition rate, while the use of slant normalization based on contextual information allowed an improvement of 11.3%.

The strategies used to deal with size variation have also taken into account the implicit segmentation-based method of the SCB stage. The experiments have shown that training the numeral HMMs taking into account the

intra-string size variation is more promising than using a non-linear size normalization method. With this contextual information during training of the numeral models, we avoid the possible distortions on the digit strokes caused by a non-linear size normalization method. These experiments have shown that the use of size normalization brought an improvement of 1.6% on the recognition performance, while the use of IntraSSV as contextual information brought about an improvement of 4%.

The HMM topology with an end-state has ensured a more precise definition of the string segmentation cuts by the LBA. The experiments have shown that although it does not bring about a significant improvement in the recognition of isolated numerals (about 0.7%), it did bring about an improvement of 7.1% in the global string recognition rate. This additional state also enabled the use of a two-step training mechanism to incorporate further string contextual information in the numeral models. We can observe that the space model brought about some improvement in terms of recognition performance for each string class. Even without such significant results, these experiments have shown that an investigation on modeling other kinds of interactions between adjacent digits, such as touching and overlapping, may be a promising way of obtaining an accurate string representation.

The scheme used for optimizing the HMM parameters of the numeral models in the SCB stage provides a further improvement of 2.8% in the global string recognition rate. In this optimization scheme, we have redefined the number of states of the numeral HMMs and also the codebook size.

After all these experiments had been conducted to construct and evaluate the SCB stage, the final recognition rates for strings composed of 2, 3, 4, 5, 6, and 10 digits were: 90.2%, 85.8%, 81.6%, 79.9%, 76.7%, and 68.4%, respectively.

The Verification stage has been shown to make a significant contribution to string recognition performance [4]. After this stage, the final recognition rate of known-length strings composed of 2, 3, 4, 5, 6, and 10 digits were: 95.2%, 92.6%, 92.1%, 90.0%, 90.0%, and 86.9%, respectively. This means an improvement on the global string recognition rate of 9.9%. Similarly, the recognition rate of touching digit pairs (TDPs) had improved from 79.5% to 89.6%.

The strategies used to consider unknown-length numeral strings showed a small loss in terms of recognition performance compared to the previous results. The most promising results were obtained using the string length predictor. The final recognition rates of strings composed of 2, 3, 4, 5, 6, and 10 digits were: 94.8%, 91.6%, 91.2%, 88.3%, 89.0%, and 86.9%, respectively. This means a loss in terms of global recognition rate of 0.9%.

Table 22 shows a comparison of the proposed method with other methods. Even considering only methods evaluated on numeral strings extracted from NIST database, the comparison with other methods is delicate in some cases because of the uncertainty concerning the exact

Table 22. Performance of numeral strings based data in NIST SD19

| Reference | String class | Recog. rate (%) | Error rate (%) | Reject. rate (%) | # samples |
|-------------------------|----------------------|-----------------|----------------|------------------|-----------|
| Keeler & Rumelhart [12] | 2_digit | 87.0 | 1.0 | 12.0 | 1,000 |
| | 3_digit | 84.0 | 1.0 | 15.0 | 1,000 |
| | 4_digit | 76.0 | 1.0 | 23.0 | 1,000 |
| | 5_digit | 71.0 | 1.0 | 28.0 | 1,000 |
| | 6_digit | 62.0 | 1.0 | 37.0 | 1,000 |
| Matan et al. [21] | 10_digit | NA | NA | NA | NA |
| | 2_digit | 94.2 | 1.0 | 4.8 | 1,000 |
| | 3_digit | 87.9 | 1.0 | 11.1 | 1,000 |
| | 4_digit | 79.9 | 1.0 | 19.1 | 1,000 |
| | 5_digit | 75.6 | 1.0 | 23.4 | 1,000 |
| Ha et al. [10] | 6_digit | 63.3 | 1.0 | 35.7 | 1,000 |
| | 10_digit | NA | NA | NA | NA |
| | 2_digit | 96.2 | 3.8 | 0.0 | 981 |
| | 3_digit | 92.7 | 7.3 | 0.0 | 986 |
| | 4_digit | 93.2 | 6.8 | 0.0 | 988 |
| Oliveira et al. [24] | 5_digit | 91.1 | 8.9 | 0.0 | 988 |
| | 6_digit | 90.3 | 9.7 | 0.0 | 982 |
| | 10_digit | NA | NA | NA | NA |
| | 2_digit | 95.6 | 4.4 | 0.0 | 2,370 |
| | 3_digit | 93.0 | 7.0 | 0.0 | 2,385 |
| Procter & Elms [26] | 4_digit | 90.2 | 9.8 | 0.0 | 2,345 |
| | 5_digit | 89.0 | 11.0 | 0.0 | 2,316 |
| | 6_digit | 88.5 | 11.5 | 0.0 | 2,169 |
| | 10_digit | 84.8 | 15.2 | 0.0 | 1,217 |
| | 2,...,6 and 10_digit | 74.2 | 25.8 | 0.0 | 1,400 |
| Proposed method | 2_digit | 94.8 | 5.2 | 0.0 | 2,370 |
| | 3_digit | 91.6 | 8.4 | 0.0 | 2,385 |
| | 4_digit | 91.3 | 8.7 | 0.0 | 2,345 |
| | 5_digit | 88.3 | 11.7 | 0.0 | 2,316 |
| | 6_digit | 89.1 | 10.9 | 0.0 | 2,169 |
| 10_digit | 86.9 | 13.1 | 0.0 | 1,217 | |

Table 23. Performance of touching digit pairs (TDPs) extracted from NIST SD19

| Reference | Recog. rate (%) | Error rate (%) | Reject. rate (%) | # samples |
|------------------|-----------------|----------------|------------------|-----------|
| Chi et al. [5] | 89.2 | 10.8 | 0.0 | 3,355 |
| Yu & Yan [34] | 89.7 | 10.3 | 0.0 | 3,355 |
| Zhou et al. [35] | 85.7 | 3.5 | 10.8 | 4,395 |
| Proposed method | 89.6 | 10.4 | 0.0 | 2,069 |

data being used and the different number of samples. The comparison with the method proposed by Oliveira et al. [24] is more interesting, since the authors have used the same testing set. Table 23 presents a comparison for touching digit pairs.

5 Conclusions and future work

We have described a two-stage HMM-based method for recognizing handwritten numeral strings. With the first stage, we showed that the use of an implicit segmentation strategy is a promising way to deal with the string

difficulties. The reason is that it avoids the need to define heuristics to group parts of broken digits or to separate touching digits, such as those used in the segmentation-based methods. However, there is some cost attached to this strategy related to the loss in terms of recognition performance caused by joining segmentation and recognition processes. During the experiments, it was possible to observe that the feature set and numeral models defined in the first stage (SCB), which have often been shown to be capable of finding the right segmentation points, are not strong enough in terms of recognition performance. With the second stage, we showed the contribution to handwritten numeral string recognition performance of considering a further verification step, which is used to re-rank the hypotheses of the first stage. Verification compensates for the loss in terms of recognition performance resulting from the necessary tradeoff between segmentation and recognition carried out in the implicit segmentation method.

This two-stage method has enabled the use of two different feature sets and numeral models: one taking into account both segmentation and recognition aspects in an implicit segmentation-based strategy, and the other considering only the recognition aspects of isolated digits. In other words, the Verification stage is used to complement the SCB stage, in the sense that their features and numeral models are strong in terms of recognition performance.

We may improve the performance of the proposed method by further development in a number of areas. A simple improvement would be to develop a rejection mechanism. In addition, we can improve the performance of the proposed method by further investigating feature sets, since this method enables the combination of different features at each stage. For instance, a new set of foreground features can be defined to improve the segmentation-recognition performance of the first stage, while new features with powerful recognition performance can be evaluated in the second stage. In addition, the construction of the codebooks must receive more attention – different vector quantization processes and codebook lengths should be evaluated.

In a similar way, further work can be carried out to improve the numeral models of each stage. For instance, in the first stage, it would be interesting to investigate a way of integrating additional contextual information into the numeral models regarding the interaction between adjacent numerals in strings.

A final consideration regarding the computing time. We did not take into account the computing time during the experiments. However, the N -best algorithm has been evaluated as the most time-consuming module of the proposed method. To minimize this problem, the technique proposed in [14] can be evaluated.

Acknowledgements. The authors wish to thank the Pontifícia Universidade Católica do Paraná (PUC-PR, Brazil), the Universidade Estadual de Ponta Grossa (UEPG, Brazil), the École de Technologie Supérieure (ETS, Canada), the Cen-

tre for Pattern Recognition and Machine Intelligence (CEN-PARMI, Canada), and the Fundação Araucária (Paraná, Brazil), which have supported this work.

References

1. Bose CB, Kuo SS (1994) Connected and degraded text recognition using Hidden Markov Model. *Pattern Recognition* 27(10):1345–1363
2. Britto AS, Sabourin R, Lethelier E, Bortolozzi F, Suen CY (2000) Improvement in handwritten numeral string recognition by slant normalization and contextual information. In: *Proc. 7th International Workshop on Frontiers on Handwriting Recognition*, September 11–13, Amsterdam, The Netherlands, 1:323–332
3. Britto AS, Sabourin R, Bortolozzi F, Suen CY (2001) An enhanced HMM topology in an LBA framework for the recognition of handwritten numeral strings. In: *Proc. International Conference on Advances in Pattern Recognition*, 1:105–114, Rio de Janeiro, Brazil
4. Britto AS, Sabourin R, Bortolozzi F, Suen CY (2001) A two-stage HMM-based systems for recognizing handwritten numeral strings. In: *Proc. International Conference on Document Analysis and Recognition*, pp 396–400, Seattle, USA
5. Chi Z, Sutens M, Yan H (1995) Separation of single and double touching handwritten numeral strings. *Opt Eng* 34:1159–1165
6. Elms AJ, Procter S, Illingworth J (1998) The advantage of using an HMM-based approach for faxed word recognition. *Int J Doc Anal Recognition*:18–36
7. Fujisawa H, Nakano Y (1992) Segmentation methods for character recognition: from segmentation to document structure analysis. In: *Proc. IEEE* 80:1079–1092
8. Gader PD, Keller JM, Krishnapuram R, Chiang J, Mohamed MA (1997) Neural and fuzzy methods in handwriting recognition. *Comput Mag* :79–85
9. Guillevic D, Suen CY (1997) HMM Word recognition engine. In: *Proc. 4th International Conference on Document Analysis and Recognition*, pp 544–547
10. Ha TM, Zimmermann M, Bunke H (1998) Offline handwritten numeral string recognition by combining segmentation-based and segmentation-free methods. *Pattern Recognition* 31(3):257–272
11. Huang XD, Ariki Y, Jack MA (1993) Hidden Markov Models for speech recognition. *Edinburgh Information Technology Series*, Edinburgh University Press, Edinburgh, UK, pp 276
12. Keeler J, Rumelhart DE (1992) A self-organizing integrated segmentation and recognition neural network. In: *Moody JE, Hanson SJ, Lippmann RP (eds) Advances in neural information processing systems, vol 4*. Morgan Kaufmann, San Mateo, Calif., USA, pp 496–503
13. Kimura F, Inoue S, Wakabayashi T, Tsuruoka S, Miyake Y (1998) Handwritten numeral recognition using autoassociative neural networks. In: *Proc. 14th International Conference on Pattern Recognition* pp 166–171
14. Koerich AL, Sabourin R, Suen CY (2001) A time-length constrained level building algorithm for large vocabulary handwritten word recognition. In: *Proc. 2nd International Conference on Advances on Pattern Recognition*
15. Lee SW, Kim SY (1999) Integrated segmentation and recognition of handwritten numerals with cascade neural network. *IEEE Trans Syst Man Cybern* 29(2):285–290
16. Lee SW, Park JS (1994) Nonlinear shape normalization methods for the recognition of large-set handwritten characters. *Pattern Recognition* 27(7):895–902
17. Linde Y, Buzo A, Gray RM (1980) An algorithm for vector quantization design. *IEEE Trans Comm* 28:84–95
18. Makhoul J, Roucos S, Gish H (1985) Vector quantization in speech coding. In: *Proc. IEEE* 73:1551–1558
19. Makhoul J, Schwartz R, Lapre C, Bazzi I (1998) A script-independent methodology for optical character recognition. *Pattern Recognition* 31(9):1285–1294
20. Matan O, Burges Y, Cum L, Denker JS (1992) Multi-digit recognition using a space displacement neural network. In: *Moody JE, Hanson SJ, Lippmann RP (eds) Advances in neural information processing systems, vol 4*. Morgan Kaufmann, San Mateo, Calif., USA, pp 488–495
21. Matin GL, Rashid M, Pittman JA (1993) Integrated segmentation and recognition through exhaustive scans or learned saccadic jumps. *Int J Pattern Recognition Artif Intell* 7(4):831–847
22. Nishida H, Mori S (1994) A model-based split-and-merge method for character string recognition. *Int J Pattern Recognition Artif Intell* 8(5):1205–1222
23. Oliveira LS, Lethelier E, Bortolozzi F, Sabourin R (2000) A new approach to segment handwritten digits. In: *Proc. 7th International Workshop on Frontiers on Handwriting Recognition*, Sept. 11–13, Amsterdam, The Netherlands 1:577–582
24. Oliveira LS, Lethelier E, Bortolozzi F, Sabourin R (2001) A modular system to recognize numerical amounts on Brazilian bank cheques. In: *Proc. International Conference on Document Analysis and Recognition* pp 389–394, Seattle, USA
25. Oliveira LS, Sabourin R, Bortolozzi F, Suen CY (2002) A recognition and verification strategy. *IEEE Trans Pattern Anal Mach Intell* (accepted for publication)
26. Procter S, Elms AJ (1998) The recognition of handwritten digit strings of unknown length using Hidden Markov Models. In: *Proc. 14th International Conference on Pattern Recognition* pp 1515–1517
27. Rabiner LR (1989) A tutorial on Hidden Markov Models and selected applications in speech recognition. In: *Proc. IEEE* 77(2):257–286
28. Rabiner LR, Juang BH (1993) *Fundamentals of speech recognition*. Prentice-Hall, Englewood Cliffs, N.J., USA
29. Sabourin R (1990) *Une Approche de Type Compréhension de Scene Appliquée au Probleme de la Vérification Automatique de L'Identité par L'Image de la Signature Manuscrite*. These de Doctorate, Departement de Génie Électrique, École Polytechnique, Université de Montreal
30. Shi Z, Srihari N, Shin YC, Ramanaprasad V (1997) A system for segmentation and recognition of totally unconstrained handwritten numeral strings. In: *Proc. 4th International Conference on Document Analysis and Recognition* 2:455–458
31. Suen CY, Nadal C, Legault R, Mai TA, Lam L (1992) Computer recognition of unconstrained handwritten numerals. In: *Proc. IEEE* 80(7):1162–1180
32. Wang X (1994) Durationally constrained training of HMM without explicit state duration PDF. In: *Proc.*

Institute of Phonetic Sciences, University of Amsterdam, 18:111–130

33. Westall JM, Narasimha MS (1993) Vertex-directed segmentation of handwritten numerals. *Pattern Recognition* 26(10):1473–1486
34. Yu D, Yan H (1998) Separation of single-touching handwritten numeral strings based on structural features. *Pattern Recognition* 31(12):1835–1847
35. Zhou J, Suen CY (2000) Recognition and verification of touching handwritten numerals. In: *Proc. 7th International Workshop on Frontiers on Handwriting Recognition*, September 11–13, Amsterdam, The Netherlands, 1:179–188



Alceu de Souza Britto Jr. received M.Sc. degree in Industrial Informatic from the Federal Center for Technological Education of Paraná (Brazil) in 1996, and Ph.D. degree in Computer Science from Pontifical Catholic University of Paraná (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 PUC-PR. From July of 1998 to July of 2000 he worked on handwriting recognition area in Montreal (Canada) at the laboratories of the Centre for Pattern Recognition and Machine Intelligence (CENPARMI-Concordia University) and the École de technologie supérieure (ÉTS-Université du Québec). His research interests are in the areas of document analysis and handwriting recognition.



Robert Sabourin received B.ing, M.Sc.A, Ph.D. degrees in electrical engineering from the École Polytechnique de Montréal in 1977, 1980 and 1991, respectively. In 1977, he joined the physics department of the Université de Montréal where he was responsible for the design and development of scientific instrumentation for the Observatoire du Mont Mégantic. In 1983, he joined the staff of the École de Technologie Supérieure, Université

du Québec, Montréal, P.Q, Canada, where he is currently a professeur titulaire in the Département de Génie de la Production Automatisée. In 1995, he joined also the Computer Science Department of the Pontificia Universidade Católica do Paraná (PUC-PR, Curitiba, Brazil) where he was co-responsible since 1998 for the implementation of a PhD 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.



Flávio Bortolozzi obtained the B.S. degree in mathematics in 1977 from Pontifícia Universidade Católica do Paraná (PUC-PR), Brazil, a B.S. degree in civil engineering in 1980 from PUC-PR, and a Ph.D. degree in system engineering (computer vision) from the Université de Technologie de Compiègne, France, in 1990 where his work was concerned on the trinocular vision. From 1994 to 1999 he was the

head of the department of informatics, and the dean of the college of exact sciences and technology at PUC-PR. Currently, he is a full professor of computer science department and the pro-rector for research at PUC-PR. His research interests are computer vision, handwriting recognition, document image analysis, educational multimedia, and hypermedia.



Ching Y. Suen received an M.Sc.(Eng.) degree from the University of Hong Kong and a Ph.D. degree from the University of British Columbia, Canada. In 1972, he joined the Department of Computer Science of Concordia University where he became Professor in 1979 and served as Chairman from 1980 to 1984, and as Associate Dean for Research of the Faculty of Engineering and Computer

Science from 1993 to 1997. Currently he holds the distinguished Concordia Research Chair of Artificial Intelligence and Pattern Recognition, and is the Director of CENPARMI, the Centre for PR & MI. Prof. Suen is the author/editor of 11 books and more than 300 papers on subjects ranging from computer vision and handwriting recognition, to expert systems and computational linguistics. He is the founder and Editor-in-Chief of a journal and an Associate Editor of several journals related to pattern recognition. A Fellow of the IEEE, IAPR, and the Academy of Sciences of the Royal Society of Canada, he has served several professional societies as President, Vice-President, or Governor. He is also the founder and chair of several conference series including ICDAR, IWFHR, and VI. Currently he is the General Chair of the International Conference on Pattern Recognition to be held in Quebec City in August 2002. Dr. Suen is the recipient of several awards, including the ITAC/NSERC Award in 1992 and the Concordia “Research Fellow” award in 1998.