Automatic Recognition of Handwritten Numerical Strings: A Recognition and Verification Strategy

Luiz S. Oliveira, Robert Sabourin, *Member*, *IEEE*, Flávio Bortolozzi, and Ching Y. Suen, *Fellow*, *IEEE*

Abstract—A modular system to recognize handwritten numerical strings is proposed. It uses a segmentation-based recognition approach and a Recognition and Verification strategy. The approach combines the outputs from different levels such as segmentation, recognition, and postprocessing in a probabilistic model. A new verification scheme which contains two verifiers to deal with the problems of oversegmentation and undersegmentation is presented. A new feature set is also introduced to feed the oversegmentation verifier. A postprocessor based on a deterministic automaton is used and the global decision module makes an accept/reject decision. Finally, experimental results on two databases are presented: numerical amounts on Brazilian bank checks and NIST SD19. The latter aims at validating the concept of modular system and showing the robustness of the system using a well-known database.

Index Terms—Handwritten numerical string recognition, segmentation and recognition of numerals, recognition and verification, feature extraction, probabilistic model.

1 Introduction

Automatic reading of numerical fields has been attempted in several application areas. One such area is the reading of courtesy amounts on bank checks. This application has been very popular in handwriting recognition research, due to the availability of relatively inexpensive CPU power, and the possibility to reduce considerably the manual effort involved in this task. Another application is the reading of postal zip codes in addresses written or typed on envelopes. The former is more difficult than the latter due to a number of differences in the nature of the handwritten material. For example, bank check systems have to take into account the great variability in the representation of a numerical amount, e.g., the number of components to be identified, which is not necessary for a zip code system since the number of digits is fixed and known a priori. Another important requirement from a bank check system is its reliability. It has been estimated that such a system becomes commercially efficient only when the error rate is 1 percent or lower.

Methods for courtesy amount recognition belong to the class of digit recognition techniques although in many cases these amounts include also some nondigit symbols such as commas, periods, strokes, currency names, etc. Strategies for digit string recognition can be divided into segmentation-then-recognition [30] and segmentation-based recognition [26]. In the first approach, the segmentation module provides a single sequence hypothesis where each subsequence should contain an isolated character which is submitted to the recognizer. This technique shows its limits rapidly when the correct segmentation does not fit with the predefined rules of the segmenter. Very often, contextual information is used during the segmentation process to improve the robustness of the system.

The second strategy is based on a probabilistic assumption where the final decision must express the best segmentation-recognition score of the input image. Usually, the system yields a list of hypotheses from the segmentation module and each hypothesis is then evaluated by the recognition. Finally, the list is postprocessed taking into account the contextual information. Although this approach gives a better reliability than the previous one, the main drawback lies in the computational effort needed to compare all the hypotheses generated. Moreover, the recognition module has to discriminate various configurations such as fragments, isolated characters, and connected characters. In this strategy, segmentation can be explicit when based on cut rules [27], [7] or implicit when each pixel column is a potential cut location [5], [22].

In this paper, we present a modular recognition system for handwritten numerical strings. This system takes a segmentation-based recognition approach where an explicit segmentation algorithm determines the cut regions and provides a multiple spatial representation. Contrary to the systems that process just isolated numerals [17], [13], our system has to solve a crucial problem: distinguishing, at the recognition stage, a sequence corresponding to an intercharacter segmentation from another relative to an intracharacter segmentation. In order to deal with this problem, we have proposed a strategy based on Recognition and Verification where the recognition function takes into account only a general-purpose recognizer, while the verifiers evaluate the

- *L.S. Oliveira and R. Sabourin are with École de Technologie Supérieure, Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle (LIVIA), 1100, rue Notre Dame Ouest, Montreal, QC, Canada, H3C 1K3 and the Centre for Pattern Recognition and Machine Intelligence (CENPARMI), Concordia University, 1455 de Maisonneuve Blvd. West, Suite GM 606, Montreal, Canada, H3G 1M8.*
  E-mail: soares@cenparmi.concordia.ca, sabourin@gpa.etsmtl.ca.
- *F. Bortolozzi and R. Sabourin are with the Programa de Pós-Graduacão em Informática Aplicada (PPGIA), Pontíficia Universidade Católica do Paraná, Rua Imaculada Conceição 1155, Prado Velho, 80215-901, Curitiba, PR, Brazil. E-mail: fborto@ppgia.pucpr.br.*
- *C.Y. Suen is with the Centre for Pattern Recognition and Machine Intelligence (CENPARMI), 1455 de Maisonneuve Blvd. West, Suite GM 606, Montreal, Canada, H3G 1M8. E-mail: suen@cenparmi.concordia.ca.*

Manuscript received 23 Apr. 2001; revised 19 Nov. 2001; accepted 2 Apr. 2002. Recommended for acceptance by L. Vincent.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 114038.

result produced by the recognizer. The integration of all modules is done through a probabilistic model inspired by information theory [20].

The focus of this work is to show how the verification modules can improve the recognition rate and reliability of the system. We introduce a new scheme of verification where two verifiers are considered. The first one deals with oversegmentation, while the second one deals with undersegmentation. A new feature set, which takes into account multilevel concavity analysis and contextual information, was developed to feed the oversegmentation verifier. We present also the concept of modular recognition system and we show how such a recognition system can deal with different applications. We will see in the experimental results that such a strategy of verification clearly improves the overall performance of the system. In order to validate such a concept and to evaluate the robustness of the system, we present experiments on two different databases: numerical amounts on Brazilian bank checks and digit strings on NIST SD19.

This paper is structured as follows: Section 2 provides a brief overview of the system. Section 3 presents the probabilistic model on which our system is based, definitions related to the modularity of the system, and levels of verification as well. Section 4 describes the overall architecture of the system. Section 5 reports the experiments carried out and Section 6 includes some discussion and comparison. Finally, Section 7 presents our conclusions.

2 System Overview

The system discussed in this work takes a segmentation-based recognition approach and a Recognition and Verification strategy. The outputs from different modules of the system such as segmentation, recognition, and postprocessing are combined using a probabilistic framework (described in Section 3.1), which adds a degree of tractability to understand the interactions among the system modules.

An explicit segmentation algorithm determines the cut regions and provides multiple spatial representation. After segmentation, different kinds of features are extracted in order to feed the recognition module, which is composed of one general-purpose recognizer and two verifiers. The goal of the verifiers is to improve the overall performance of the system by detecting over and undersegmentation. All classifiers used in the system are described in Section 3.2. Thereafter, the system generates a list of hypotheses through a modified Viterbi algorithm and then each hypothesis is syntactically analyzed by means of a deterministic automaton. Finally, the global decision module makes an accept/rejection decision.

Despite the fact that the verifiers presented in this work are classifiers with probabilistic outputs, they are called verifiers because they do not play the same role as the general-purpose recognizer does in the system. In Section 3.3, we discuss the verification and its different levels as well. In Section 3.4, we present the idea of modular system employed in this work. Finally, Section 4 describes all system modules and the interaction between the general-purpose recognizer and verifiers.

3 Definitions

3.1 Probabilistic Model

The goal of the probabilistic model is to define a function that combines all the system modules in order to allow a sound integration of all knowledge sources used to infer a plausible interpretation. The probabilistic model that we are using has been applied to speech recognition [24], handwritten word recognition [6], and handwritten digit recognition [23]. Such a model estimates the most probable interpretation of the written amount M (noted $\widehat{M}$). Its input corresponds to an image I after preprocessing.

In a probabilistic framework, $\widehat{M}$ is given by the maximum posterior probability:

$$P\left(\widehat{M}/I\right) = \max_{M} P(M/I). \qquad (1)$$

We consider that the amount M can be expressed by the writer through some variant V derived from the different ways of expressing M. Each variant is composed of a sequence of characters: $V = v_1, \ldots, v_m$. In this way, we can represent also the effects caused by the preprocessing. For example, 10000 is a variant of 100,00 because the comma in 100,00 may be eliminated by the preprocessing. The amount M itself will be a sequence of characters $M = m_1, \ldots, m_p$ representing what we call the canonical form of M. The decomposition of the image of variant V into elementary segments allows the introduction of a segmentation term $S = s_1, \ldots, s_m$, where m corresponds to the number of characters in V. S describes for each character of V the types of segmentation rules that may consider it to be composed of some fragments. Therefore, $s_k = r_{b_{(k)}}, \ldots, r_{e_{(k)}}$, where $r_{b_{(k)}}$ is the initial segmentation rule (b means begin) and $r_{e_{(k)}}$ is the final segmentation rule (e means end) used to generate the fragment s_k.

By summing up all segmentations and all variants, we can write

$$P(M/I) = \sum_{S}\sum_{V} P(M, V, S/I). \qquad (2)$$

Considering an approximation where a sum of probabilities over segmentations representing the same amount is replaced by the maximum probability of a single segmentation, we have:

$$\sum_{S}\sum_{V} P(M, V, S/I) \approx \max P(M, V, S/I). \qquad (3)$$

This approximation was checked numerically through experimentation on the NIST SD19 database. We have used about 2,000 images of strings of digits and we verified that such an approximation is plausible. Thus, we can assume that the image is analyzed by only one segmentation and one variant, so that

$$\exists V, S \quad \text{such that} \quad P(M/I) \approx P(M, V, S/I). \qquad (4)$$

Using Bayes rule, we can write

$$P(M/I) \approx \frac{P(I/M, V, S) \times P(M, V, S)}{P(I)}. \qquad (5)$$

The probability of amount M is then:

$$P(M/I) \propto P(I/M, V, S) \times P(S/M, V) \times P(M, V). \qquad (6)$$

TABLE 1
Description of the Classifiers Used in Our System

Classifier	Classes Recognized	Usage
e_3	3 non-numerical classes {"//", ",", "."}	To feed e_{13}
e_{10}	10 numerical classes {0..9}	To feed e_{13} (numerical amounts) and classification (NIST)
e_{13}	13 classes $\{0..9\} \cup$ {"//", ",", "."}	Classification (numerical amounts)
v_o	isolated and over-segmented characters	Verification
v_u	isolated and under-segmented characters	Verification

We assume that segmentation S only depends on M through variant V since S describes how V (and not M) is mapped onto elementary segments in I:

$$P(M/I) \approx P(I/V,S) \times P(S/V) \times P(M,V). \tag{7}$$

This equation makes explicit the terms to be estimated: the recognition $P(I/V,S)$, the segmentation term $P(S/V)$, and the joint probability of amounts and their variants $P(M,V)$.

We also assume that the shapes of digits and other symbols only depend on their class and their segmentation configuration. This is an approximation since characters are written by only one writer and because some ligatures cause contextual effects in the shapes of characters. It is important to remark that, in our system, the union of all segments is equal to the whole image and also that the system does not allow intersection between segments unless characters overlap. However, overlaps occur only rarely. In this way, we can derive the joint probability $P(I/V,S)$ from individual terms through:

$$P(I/V,S) \approx \prod_{i,k} P(i_{b_{(k)}} \dots i_{e_{(k)}}/v_k, s_k = r_{b_{(k)}} \dots r_{e_{(k)}}), \tag{8}$$

where $i_{b_{(k)}}$ is the initial segment of the image and $i_{e_{(k)}}$ is the final segment of the image.

We assume that the shape of each character does not depend on the way that it is segmented from its neighbors, i.e., character shapes are not significantly affected by the segmentation rules. Moreover, even if they are slight, the improvement that we may expect from injecting segmentation information into the model to estimate character probabilities, is balanced by the increase of the number of model parameters needed to estimate these probabilities. So that:

$$P\Big(i_{b_{(k)}} \dots i_{e_{(k)}}/v_k, s_k = r_{b_{(k)}} \dots r_{e_{(k)}}\Big) \approx P\Big(\Big[i_{b_{(k)}} \dots i_{e_{(k)}}\Big]/v_k\Big), \tag{9}$$

By applying Bayes rule, we have the following approximation:

$$P(M/I) \approx \prod_{i,k} \frac{P\Big(v_k/\Big[i_{b_{(k)}} \dots i_{e_{(k)}}\Big]\Big) \times P\Big(\Big[i_{b_{(k)}} \dots i_{e_{(k)}}\Big]\Big)}{P(v_k)} \times P(S/V) \times P(M,V), \tag{10}$$

where $P(v_k/[i_{b_{(k)}} \dots i_{e_{(k)}}])$ represents the result supplied by the recognizer, $P([i_{b_{(k)}} \dots i_{e_{(k)}}])$ are a priori probabilities of the images to be recognized, $P(v_k)$ are a priori probabilities of classes recognized by the recognition module, $P(S/V)$ defines the probability of the fragmentation of characters in the variant considered, and $P(M,V)$ is the joint probability of amounts and their variant.

3.2 Neural Classifiers

Although many types of neural networks can be used for classification purposes [25], we opted for a Multilayer Perceptron (MLP) which is the most widely studied and used neural network classifier. Moreover, MLPs are efficient tools for learning large databases [21]. Therefore, all classifiers presented in this work are MLPs trained with the gradient descent applied to a sum-of-squares error function [2]. The transfer function employed is the familiar sigmoid function.

In order to monitor the generalization performance during learning and terminate the algorithm when there is no longer an improvement, we have used the method of cross-validation. Such a method takes into account a validation set, which is not used for learning, to measure the generalization performance of the network. During learning, the performance of the network on the training set will continue to improve, but its performance on the validation set will only improve to a point, where the network starts to overfit the training set, that the learning algorithm is terminated.

MLP is a measurement-level classifier, i.e., it attributes to each possible label a measurement value to address the degree or probability that the input sample has the label. In this work, we will interpret the measurement value as estimation of a posterior probability. In order to accurately estimate Bayesian probabilities, network output values must lie in the range (0,1) and they must sum to unity. In our MLP networks, the value of each output necessarily remains between zero and one because of the sigmoidal functions used, but the criterion used for training did not require the outputs to sum to one. Nevertheless, as shown by Richard and Lippmann in [29] the summed outputs of the MLP network are always close to one. As such, normalization techniques proposed to ensure that the outputs of an MLP network are true probabilities such as softmax [4] may be unnecessary. This is further supported by results of experiments performed by Bourlard and Morgan [3], which demonstrated that the sum of the outputs of MLP networks is near one for large phoneme-classification speech-recognition problems.

Let S be a pattern space which consists of A mutually exclusive sets $S = C_1 \cup \dots \cup C_A$, each of C_i, $i \in \Lambda = 1, \dots, A$ representing a set of specified patterns called a class. Let x be an input pattern that should be assigned to one of the A existing classes. e means the classifier and $e(x) = m^i(x) | \forall i (1 \leq i \leq A)$ means that the classifier e assigns the input x to each class i with a measurement value $m^i(x)$. This definition is used for all classifiers of the system. Table 1 describes our classifiers as well as where they are used in the system.

All networks presented in Table 1 have one hidden layer where the units of input and output are fully connected with units of the hidden layer. The number of hidden units used in this work are 70, 80, 20, 30, and 30 for e_3, e_{10}, e_{13}, v_o, and v_u, respectively. These architectures were determined empirically. The learning rate and the momentum term were set at high values in the beginning to make the weights quickly fit the long ravines in the weight space, then these parameters were reduced several times according to the number of iterations to make the weights fit the sharp curvatures.

The rule that defines how the classifier assigns an input pattern x to a class i is known as decision rule. In this work, the decision rule applied to e_{13} and e_{10} (when it is used as a general-purpose recognizer) is defined as:

$$e_{13}(x) = \max_{i \in \Lambda} m^i(x). \tag{11}$$

The goal of the verifiers v_o and v_u is to validate whether an input pattern x is an isolated character or not. We will see in Section 4.3.4 that this is achieved by multiplying three measurement values (classifier and both verifiers). Then, if the outputs of the verifiers that represent the posterior probability of an input pattern x be an isolated character (first output) are low, the output supplied by the classifier will be penalized, otherwise, it will be confirmed. For this reason, the decision rule used by the verifiers simply takes the measurement value produced in their first output, which contains the posterior probability of an input pattern x to be an isolated character. Thus, v_o shares the same decision rule of v_u, which is defined as:

$$v_u(x) = m^1(x). \tag{12}$$

3.3 Levels of Verification

The Recognition and Verification scheme looks straightforward, with a verification module embedded in the traditional classification system, which has a general-purpose recognizer only. The goal of the general-purpose recognizer is to assign a given input to one of the n existing classes of the system, while the pattern verifier assumes the role of an expert to evaluate precisely the result of the recognizer in order to compensate for its weakness due to particular training and, consequently, to make the whole system more reliable. Usually, a pattern verifier is applied after a general-purpose recognizer and it is designed to "plug and play," i.e., it is used without knowing the implementation details of the recognition modules.

Takahashi and Griffin in [31] define three kinds of verification: absolute verification for each class (Is it a "0"?), one-to-one verification between two categories (Is it a "4" or a "9"?), and verification in clustered, visually similar, categories (Is it a "0," "6," or "8"?).

In addition to these definitions, we introduce the concept of levels of verification, where two levels are considered: high-level and low-level. We define as high-level verifiers those that deal with a subset of the classes considered by the general-purpose recognizer. The goal of the verifiers at this level is to confirm or deny the hypotheses produced by the general-purpose recognizer by recognizing them [31], [32]. We define as low-level verifiers those that deal with metaclasses of the system such as characters and parts of them. The purpose of a low-level verifier is not to recognize a character, but rather to determine whether a hypothesis generated by the general-purpose recognizer is valid or not [8].

In this work, we propose two low-level verifiers to cope with the oversegmentation and undersegmentation problems. The objective of these verifiers is to validate the general-purpose recognizer hypotheses by using the following metaclasses: characters, parts of characters, and undersegmented characters. Section 4 will present more details about these verifiers.

3.4 Modular System

Since a handwritten digit string recognition system has several potential applications, it is very interesting to build a system adaptable to as many different contexts as possible. Due to the magnitude and complexity of this kind of system, they are usually divided into several modules, where each one assumes specific functions in order to facilitate the construction of the system. In order to gain a better insight of the modules in terms of a generic system, we introduce two definitions related to the system modules: Application Dependent (AD) modules and Task Dependent (TD) modules.

We define as AD those modules that should be changed (replaced or removed) to cope with another context. The best example of an AD module is the postprocessor which is usually the part of the system that has more knowledge about the application. We define as TD those modules related to the task, e.g., digit string recognition. These modules can be used for various applications. In Fig. 1, the white boxes represent TD modules while the grey boxes represent AD modules.

In Fig. 1, we can see two different versions of the same system. The first version, which considers all modules (white and gray boxes), was designed to process numerical amounts of Brazilian bank checks, while the second version, which does not consider the gray boxes, was designed to process strings of digits from the NIST database. As we can notice, through the exclusion of four AD modules (*Contextual Feature*, *Structural Features*, e_3, and e_{13}) and the modification of the *Syntactic Analysis* module, we have built a new system specialized in another context. Fig. 1 also presents the relationship between the system modules and its estimators. As we can see, the proposed system takes into account the estimators of recognition ($P(v_k/[i_{b_{(k)}} \dots i_{e_{(k)}}])$) and postprocessing ($P(M, V)$) presented in (10). Since our classifiers are based on neural networks which provide direct estimation of the posterior probabilities, we decided not to use the estimator related to a priori probabilities ($P([i_{b_{(k)}} \dots i_{e_{(k)}}])$) of the images to be recognized. We will see that the verifiers can replace this estimator successfully. Lethelier et al. in [23] use the empirical frequencies of segmentation configurations for each class in order to compute the probability of fragmentation of characters ($P(S/V)$). However, we have observed that our segmentation algorithm does not supply discriminative information about the classes of the system. This is illustrated in Fig. 2.

Fig. 2a presents the distribution of the segmentation points provided by our segmentation algorithm [27] for four different isolated digit classes (1, 4, 7, and 8) of the training set of the NIST database. We can observe that even for different classes the segmentation algorithm provides a very similar distribution of segmentation points, hence, it will be difficult to improve the performance of the system using the primitives generated by the segmentation algorithm. The second problem can be observed in Fig. 2b. In this case, the segmentation algorithm will not produce any

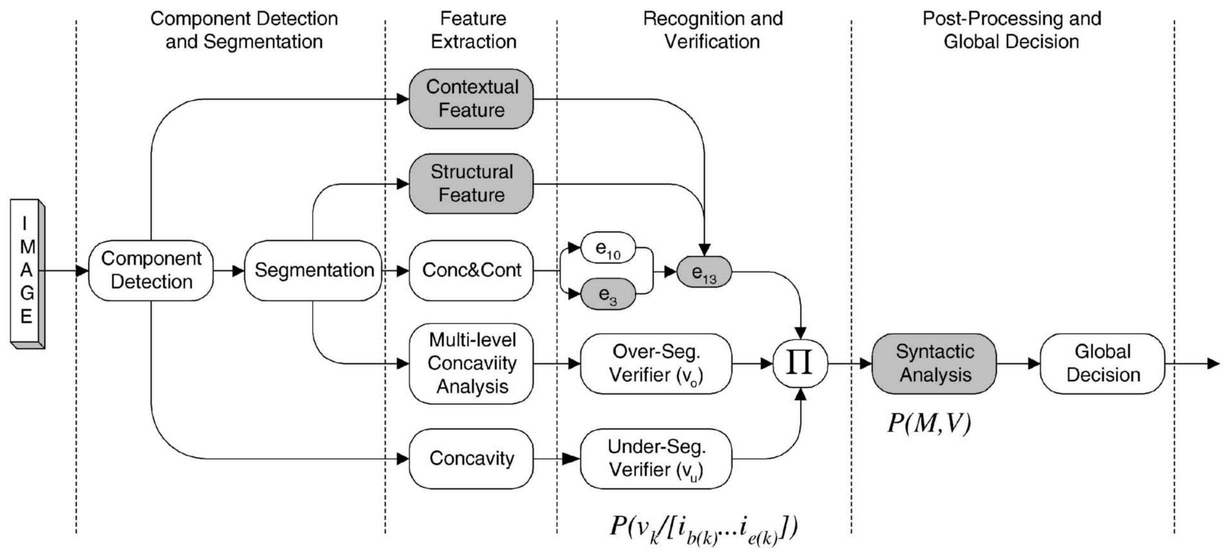


Fig. 1. Block diagram of a numeric string recognition system.

segmentation point due to the lack of minima and maxima on the superior and inferior contours, respectively. For these reasons, we choose not to use the estimator $P(S/V)$.

4 Description of the Modules

In this section, we describe all system modules depicted in Fig. 1. We are using binary images (300 dpi). The system output provides a list of hypotheses associated with a probability.

4.1 Component Detection

The goal of this module is to divide an image into groups of components, called partial images, where each group should represent an integral number of components. This allows us to convert the recognition of a string image to that of its partial images thus reducing the complexity of the subsequent tasks. This module operates in three steps: connected component analysis, delimiter detection, and grouping. The first one segments the string image into connected components and eliminates very small ones by filtering.

The second step aims at identifying specific parts of the numerical amount, namely, period (".") and comma (","). By inspecting our numerical amount database, we noticed that such components are located in the inferior part of the image. Thus, we consider as delimiters those components located entirely below the median line of the numeral string.

The last step tries to overcome the effects of fragmentation. A connected component can represent either an integer number of characters or not. The second situation is critical and should be avoided. Basically, the grouping step tries to group a character composed of several components by detecting potential parts and grouping each of them to its nearest neighbor. We have adopted a strategy similar to the one presented by Ha et al. in [14].

4.2 Segmentation

The segmentation module that we have used in this system is based on the relationship of two complementary sets of structural features, namely, contour/profile and skeletal points.

The segmentation hypotheses are generated through a segmentation graph (Fig. 3b). This initial graph is then decomposed into linear subgraphs which represent the segmentation hypotheses. Fig. 3c shows the linear subgraphs that represent the best hypothesis of segmentation. More details about this method can be found in [27].

4.3 Recognition and Verification

In this section, we describe all modules defined previously in Section 3.2. In addition, we present the feature set used by each classifier and also describe how the low-level verifiers and general-purpose recognizer interact with each other.

4.3.1 General-Purpose Recognizer

Our general-purpose recognizer is composed of three modules: e_{10}, e_3, and e_{13} (see Fig. 1). e_{10} and e_3 are

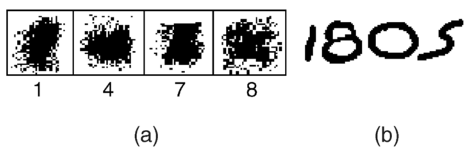


Fig. 2. Problems of using segmentation estimator: (a) Distribution of the segmentation points for isolated digit classes 1, 4, 7, and 8. (b) Samples of isolated digits where the segmentation will not provide any segmentation point.

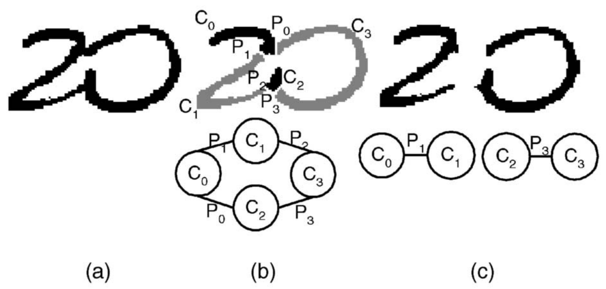


Fig. 3. Management of the segmentation: (a) Original image, (b) initial segmentation graph where P_i means the segmentation path and C_i means the subcomponent, and (c) best hypothesis of segmentation.

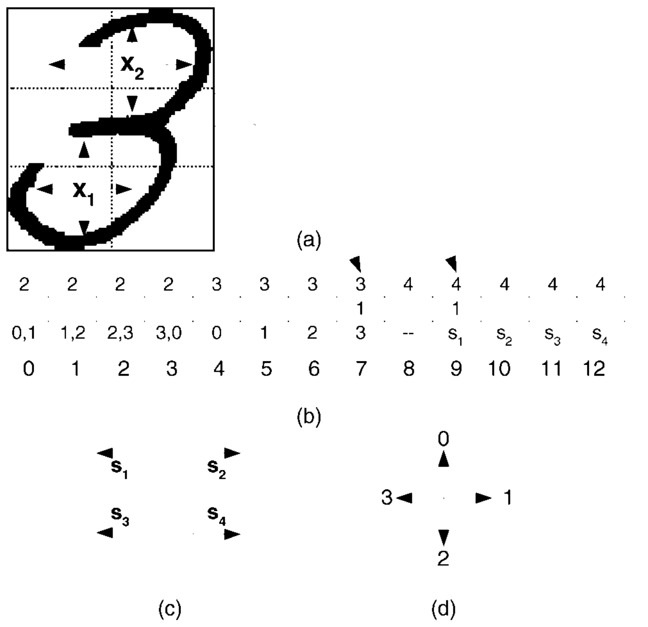


Fig. 4. Concavities measurement: (a) Concavities, (b) feature vector, (c) auxiliary directions, and (d) 4-Freeman directions.

specialized in 10 numerical and three nonnumerical ("#," ",", and ".") classes, respectively. Both classifiers use a mixture of concavity, contour-based features and surface of the characters.

The basic idea of concavity measurements [15] is the following: for each white pixel in the component, we search in 4-Freeman direction (Fig. 4d), the number of black pixels that it can reach as well as which directions the black pixel is not reached. When black pixels are reached in all directions (e.g., point x_1 in Fig. 4b), we branch out into four auxiliary directions (s_1 to s_4 in Fig. 4c) in order to confirm if the current white pixel is really inside a closed contour. Those pixels that reach just one black pixel are discarded.

Thereafter, we increment the position in the feature vector that fits with results returned by the search (Figs. 4a and 4b). In Fig. 4b, we represent the feature vector where each component has two labels. The superior label means the number of black pixels found during the search while the inferior label means the directions where the black pixels were not reached. For example, the pixel x_2 (Fig. 4a) reaches the black pixel in all directions except in direction 3. Therefore, the position 7 of the feature vector is incremented. For pixel x_1, the position 9 is incremented because it reaches the black pixel in four directions. However, using the auxiliary direction s_1 we confirm that it is not inside a closed contour. When the pixel is inside a closed contour, the position incremented is the eighth.

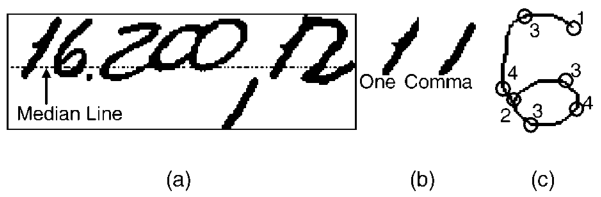


Fig. 6. Contextual and structural features: (a) Numerical amount with the median line detected, (b) similarity between digit one and symbol comma, and (c) structural features extracted of the skeleton.

Since we are dividing the image into six zones, we consider six feature vectors of 13 components each. Therefore, in the example presented above, the pixel x_2 will update the second vector while the pixel x_1 will update the fifth vector. Finally, the overall concavity feature vector is composed of (13×6) 78 components normalized between 0 and 1.

The contour information is extracted from a histogram of contour directions. For each zone, the contour line segments between neighboring pixels are grouped regarding 8-Freeman directions (Fig. 5c). The number of line segments of each orientation is counted (Fig. 5b). Therefore, the contour feature vector is composed of (8×6) 48 components normalized between 0 and 1. Finally, the last part of the feature vector is related to the character surface. We simply count the number of black pixels in each zone and normalize these values between 0 and 1. Thus, the final feature vector, which feeds e_{10} and e_3, has ($78 + 48 + 6$) 132 components.

As depicted in Fig. 1, the classifier e_{13} combines the e_{10} and e_3 outputs, one contextual feature of position, and four structural features. Therefore, e_{13} has 18 inputs. Such a scheme of combination has produced the best recognition rates in our experiments that consider the database of numerical amounts on Brazilian bank checks. The contextual feature of position, which was detected at the component detection phase, is responsible for minimizing the confusion between the digit "1" and the symbol ",". As discussed in Section 4.1, we have observed that the symbol comma is always located entirely below the median line of the numeral string. In this way, when the component is located entirely below the median line, the 14th position of the feature vector used by e_{13} receives 1, otherwise 0. As we can observe in Fig. 6, the sole information capable of removing the confusion between the digit "1" and the symbol "," is the position of the component in the image context.

In order to reduce the confusion between the numerical classes "4" and "7" and the nonnumerical symbol "#," we have used feature points of the skeleton. Four components have been considered: end points, crossing points, and two directional points, which are detected when the skeletal path changes its direction in the horizontal or vertical axis. These points are represented in Fig. 6c by the numbers 1, 2, 3, and 4,

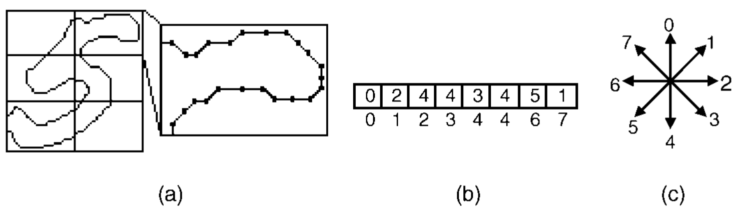


Fig. 5. Contour measurement: (a) Contour image of the upper right corner zone, (b) feature vector, and (c) 8-Freeman directions.

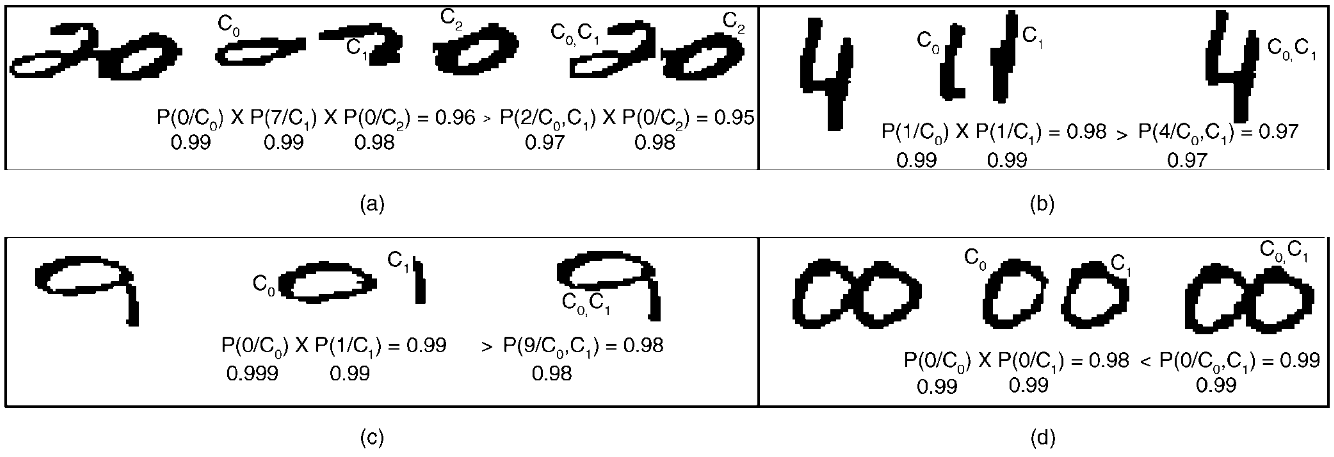

C0
C1
C2
C0,C1
C2
P(0/C0) X P(7/C1) X P(0/C2) = 0.96 > P(2/C0,C1) X P(0/C2) = 0.95
0.99 0.99 0.98 0.97 0.98
(a)
C0
C1
C0,C1
P(1/C0) X P(1/C1) = 0.98 > P(4/C0,C1) = 0.97
0.99 0.99 0.97
(b)
C1
C0
C0,C1
P(0/C0) X P(1/C1) = 0.99 > P(9/C0,C1) = 0.98
0.999 0.99 0.98
(c)
C0
C1
C0,C1
P(0/C0) X P(0/C1) = 0.98 < P(0/C0,C1) = 0.99
0.99 0.99 0.99
(d)

Fig. 7. (a), (b), and (c) Misclassification caused by oversegmentation. (d) Undersegmentation.

respectively. Once the skeletal points were detected, the points of each configuration are counted, normalized between 0 and 1, and the last four positions of the feature vector used by e_{13} updated.

We show in Section 5 that this classifier reaches very interesting recognition rates when dealing with isolated digits. However, when it is used within a complete system, it faces more complex problems such as oversegmentation and undersegmentation. In the first problem, the intracharacter segmentation hypothesis provides better results than any of the intercharacter segmentation hypotheses. Figs. 7a, 7b, and 7c show some examples of this problem. The second problem, the lack of segmentation cuts, produces better results than the correct segmentation hypothesis (Fig. 7d).

Instead of using heuristics to overcome these problems, we opted for the use of a strategy based on low-level verifiers which is more robust and reliable. Initially, we developed a strategy based on an isolated verifier which was responsible for detecting both oversegmentation and undersegmentation [28]. After some experiments, we realized that a more specialized verifier could produce better results than a generic one. In the following sections, we present both verifiers, their respective feature sets, and how they interact with the general-purpose recognizer.

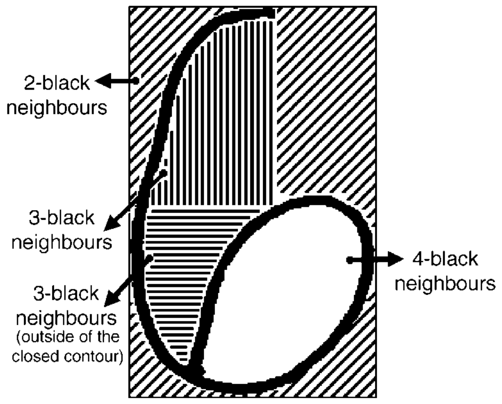


Fig. 8. Image labeled with ICL where the four possible labels are represented.

4.3.2 Oversegmentation Verifier

The main objective of this verifier is to improve the performance of the general-purpose recognizer by detecting oversegmented characters (Figs. 7a, 7b, and 7c). Thus, the verifier takes into account two classes: isolated characters and oversegmentation. In order to discriminate these two classes, we developed a new feature set called Multilevel Concavity Analysis. First of all, we introduce the definitions of *Concavity Levels*. We define as *Initial Concavity Level* (ICL) for a background pixel, the number of black neighbors that it has in the 4-Freeman directions. We also consider one label to identify the background pixels located outside of a closed contour but with four black neighbors. Those pixels that have only one black neighbor are not considered. Fig. 8 shows an image labeled with the four possible labels used in ICL.

The first step of this analysis consists of labeling with ICL the background pixels of the oversegmented piece (I_{seg}) and its corresponding original piece of handwriting (I_{orig}). Fig. 9 exhibits two examples of this procedure. Afterwards, the following verification is carried out: for each background pixel found in both images (I_{orig} and I_{seg}), we assigned to the final image (represented by the MCA in Fig. 9) a specific label (represented by the black area) when the ICLs are different, otherwise, we assigned the same ICL found in both images. The same verification procedure is carried out over the foreground pixels. However, in such a case, we assigned a specific label (represented by the bold dot) to MCA when the pixels found in both images share the same label. The latter procedure aims at computing the relative area of the oversegmented part.

The second part of this analysis concerns with I_{seg} contextual information (represented by CI in Fig. 9) which is extracted by taking into account the I_{seg} complement. As we can observe in Fig. 9, the complement that we are using is limited to I_{seg} width and it is composed of the I_{orig} background which was labeled with ICL and the surface of the I_{orig} (represented by the dot in Fig. 9). Fig. 9 also shows the final labeling which is composed of multilevel concavity analysis (MCA) and contextual information (CI) about the oversegmented piece.

Therefore, this feature set has seven possible labels: the four labels of ICL, the label that represents the difference between concavity levels of I_{orig} and I_{seg}, the label that represents the surface of the oversegmented piece, and the

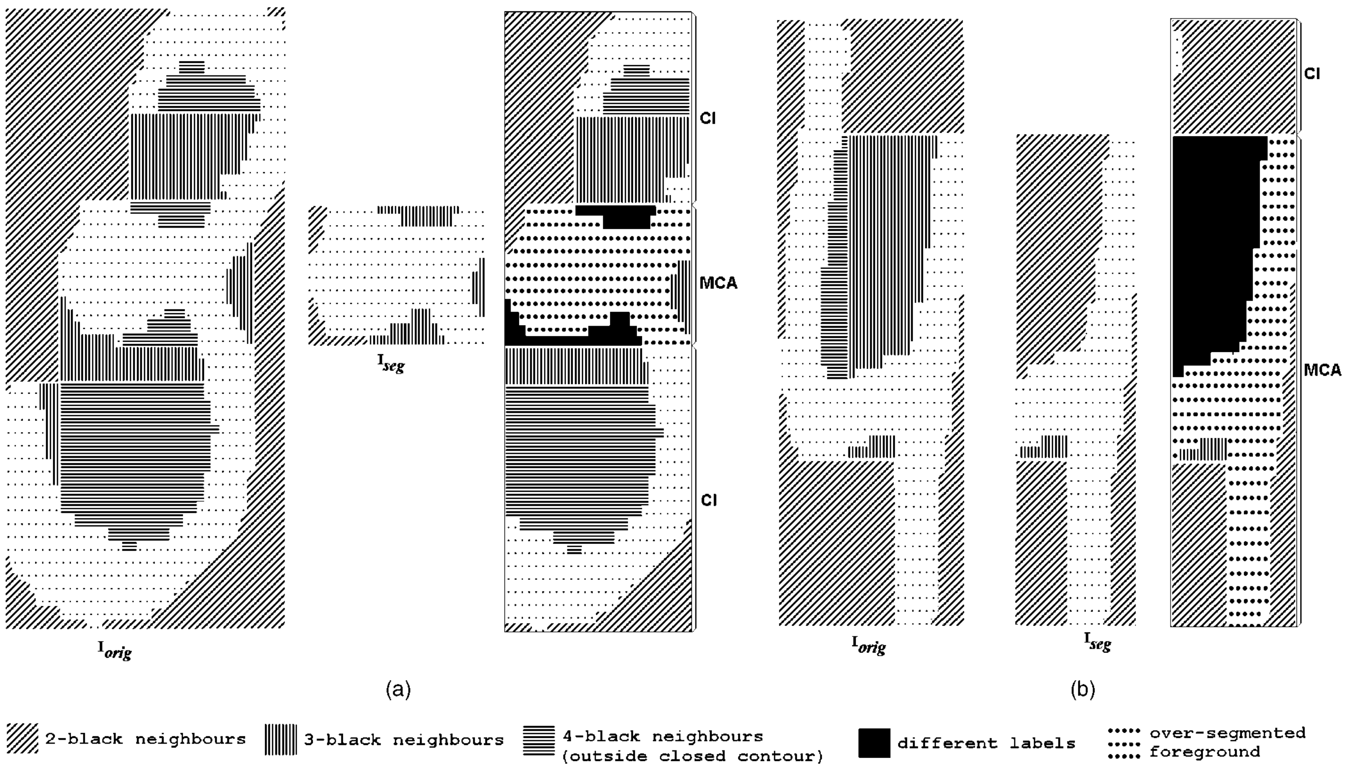


Fig. 9. (a) and (b) Two samples of Multilevel Concavity Analysis (MCA) and Contextual Information (CI).

label that represents the surface of the original image in the context of the segmentation. Finally, the multilevel concavity analysis with contextual information is divided into six regions (the same division shown in Fig. 4b) and (7×6) 42 components normalized between 0 and 1 are considered.

4.3.3 Undersegmentation Verifier

To complement the previous verifier, this one is devoted to reduce the confusion between isolated and undersegmented characters. Thus, this verifier considers two classes: isolated characters and undersegmentation.

After analyzing the system errors, we observed that touching digits with strong concavities, e.g., "00," often are recognized as "0" with a high probability. After trying some features such as, horizontal transitions, profile distances, and structural information, we conclude that the same concavity analysis and information about the surface used by the general-purpose recognizer is a good feature set to discriminate isolated characters from undersegmented ones.

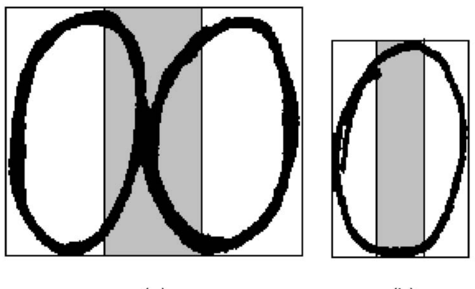


Fig. 10. Zoning used by the undersegmentation verifier.

However, a different way of zoning which divides the image into three vertical parts has been employed. Such a strategy aims at emphasizing the region of the image where the connections between characters occur often (Fig. 10). As a result, $((13 + 1) \times 3)$ 42 components normalized between 0 and 1 are considered.

4.3.4 How the Classifier and Verifiers Interact with Each Other

According to the probabilistic model presented in (10) (Section 3.1), the final probability for a hypothesis of segmentation-recognition is given through the product of the probabilities produced by its subcomponents. The probability of a subcomponent is given by the product of the probabilities produced by the general-purpose recognizer, oversegmentation verifier, and undersegmentation verifier.

In Fig. 11, we present an example of how the verifiers interact with the general-purpose recognizer. In order to better illustrate this, we reproduced the problem depicted in Fig. 7a, where the oversegmented hypothesis got a better result than the correct one. The schema in Fig. 11 is carried all the way through, from segmentation, feature extraction, to recognition, and verification. In the column "outputs," we can see the outputs of the neural networks (represented by a black box) as well as the probability produced by them. According to the decision rules presented in Section 3.2, the verifiers always consider their first output which contains a posterior probability of an input pattern of isolated character. Thus, when these probabilities are low, the output supplied by the general-purpose recognizer (e_{13}) is penalized, otherwise, it is confirmed.

We can visualize this penalty in the first two components of the second segmentation hypothesis. In such cases, the

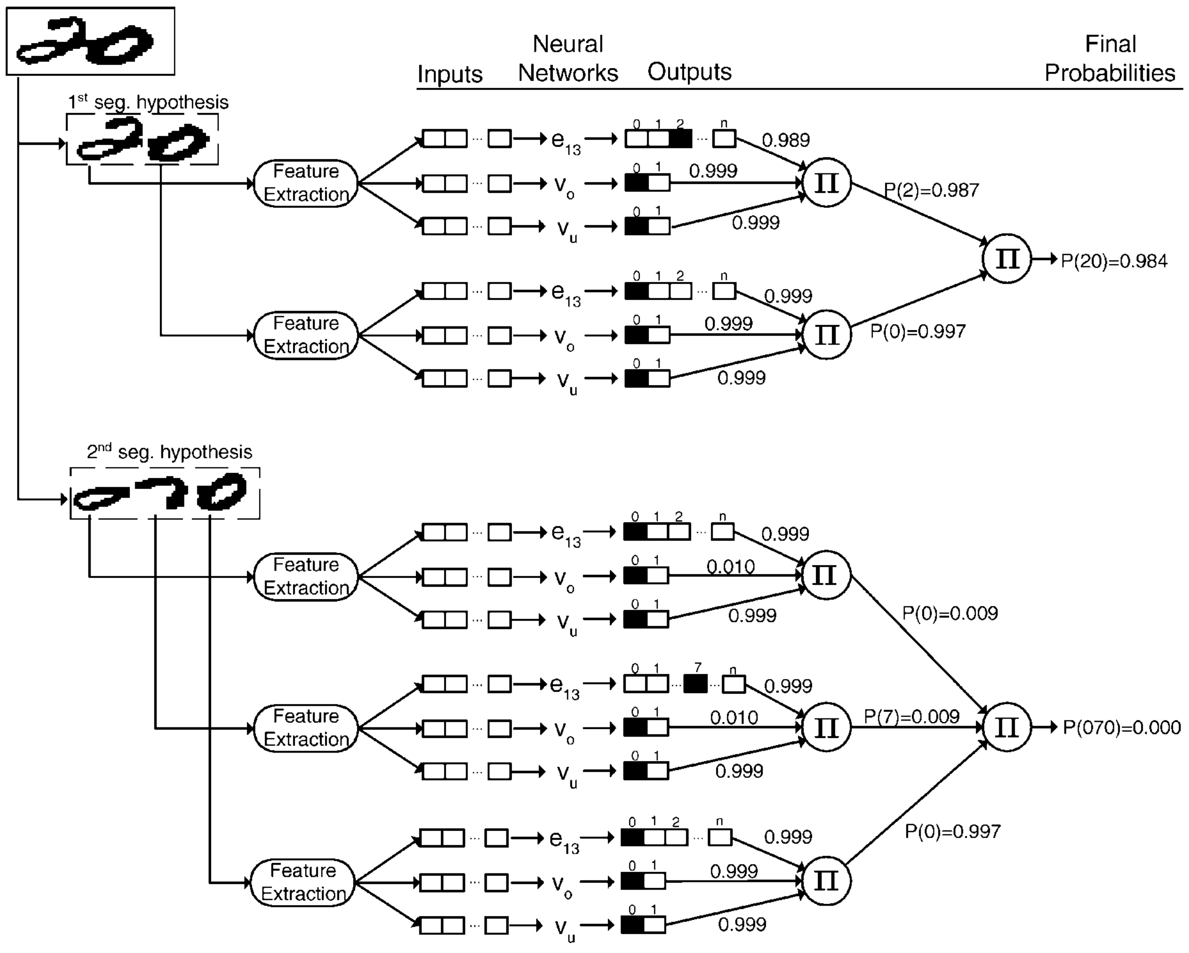


Fig. 11. Interaction between the general-purpose recognizer and verifiers.

probabilities of these components of isolated character are very low (0.010) and, hence, the probability generated by the general-purpose recognizer is penalized. The opposite happens to both components of the first segmentation hypothesis as well as the last component of the second segmentation hypothesis. In these cases, both verifiers confirmed the output produced by the general-purpose recognizer.

In the above example, we have seen how the oversegmentation verifier eliminated the confusion presented in Fig. 7a. Since the oversegmented pieces of the second segmentation hypothesis were penalized, the first segmentation hypothesis, which is the correct one, got a higher final probability.

4.4 Hypothesis Generation

The generation of k best hypotheses of an amount is carried out by means of a Modified Viterbi Algorithm [10] which ensures the calculation of the k best paths of the segmentation-recognition graph. Fig. 12 shows the three best hypotheses of segmentation-recognition.

Basically, the algorithm computes the $(i+1)\mathrm{th}$ best path with the shared subpaths calculated from rank 1 to rank i. For each node n of the graph, a stack at time t describes all the paths already computed going through this node. In order to save the predecessors for the node n at time $t-1$, pointers are also used. Thus, it is possible to show that the $(i+1)\mathrm{th}$ best path of node n at time t is conducted by one of the paths to each node at time $t-1$ driven by such pointers. When the terminal node of the graph is reached, the path is backtracked to extend the stacks and increment the pointers. Since the k best hypotheses are computed incrementally, we obtain a list of decreasing probabilities (Fig. 12c). Afterwards, each hypothesis is submitted to the postprocessor module which verifies whether it satisfies the application rules or not.

4.5 Postprocessor (pp)

In order to improve the overall performance of the system, all the hypotheses generated by the recognition module should be analyzed syntactically. Compared with the legal amount, the grammar for numerical amount is not very rich. Consequently, all the syntactic rules must be based on the nonnumerical symbols found in the numerical amount.

In spite of the fact that numerical amounts produce a poor grammar, we can find in the literature various works that use some kind of syntactic analysis. Knerr et al. in [19] consider the segments below the baseline in order to obtain the final interpretation of the numerical amount. Dimauro et al. in [9] use nonnumerical symbols such as "#" in order to identify the beginning and the end of the numerical amount. Heutte et al. in [16] present a postprocessing module for French bank checks which takes into account specific nonnumerical symbols such as the characters "F" and "C".

Usually, in Brazil, two delimiters ("#") are affixed at the beginning and at the end of the numerical amount, a period "." is sometimes used to delimit a 3-tuple of digits and a comma "," is used to identify the cents portion in the

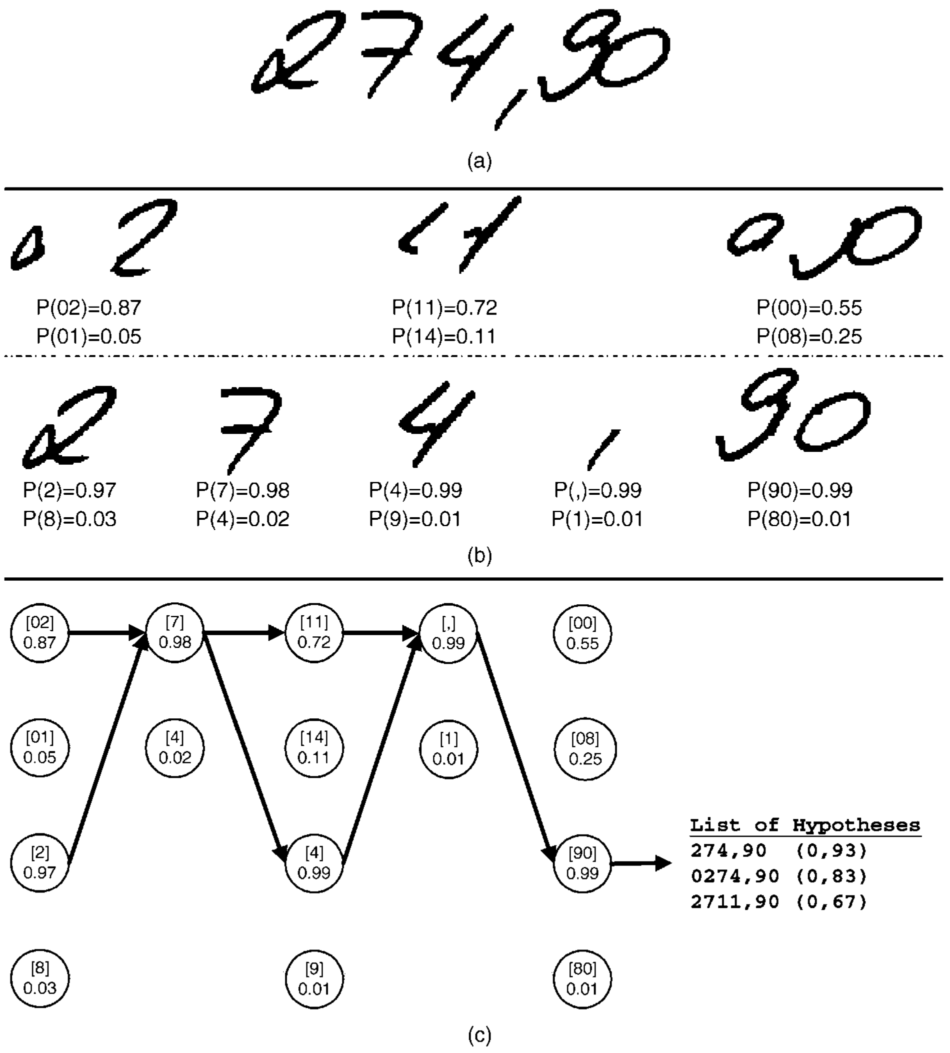


Fig. 12. Generation of global hypotheses: (a) Original image, (b) segmentation hypotheses with rank-2 hypotheses of recognition, and (c) three best paths.

numerical amount. In addition to these rules, the Central Bank of Brazil decided that the cents portion should be present in the numerical amount [1]. In order to deal with such rules we have developed a deterministic automaton which is associated with the term $P(M,V)$ of (10). This automaton is depicted in Fig. 13.

Once such an automaton is formed, we fit it to the probabilistic model in the following way: if the current hypothesis is verified by the automaton, then $P(M,V) = 0.99$, otherwise $P(M,V) = 0.01$. These values aim at reranking the list of hypotheses generated previously. Consider, for example, list of hypotheses (a) presented in Table 2, where the correct hypothesis is the second one (140,00). Since the automaton did not verify the first hypothesis of the original list, its probability will be multiplied by 0.01, while the probabilities of others will be multiplied by 0.99. List (b) of Table 2 shows the decreasing probabilities after postprocessing.

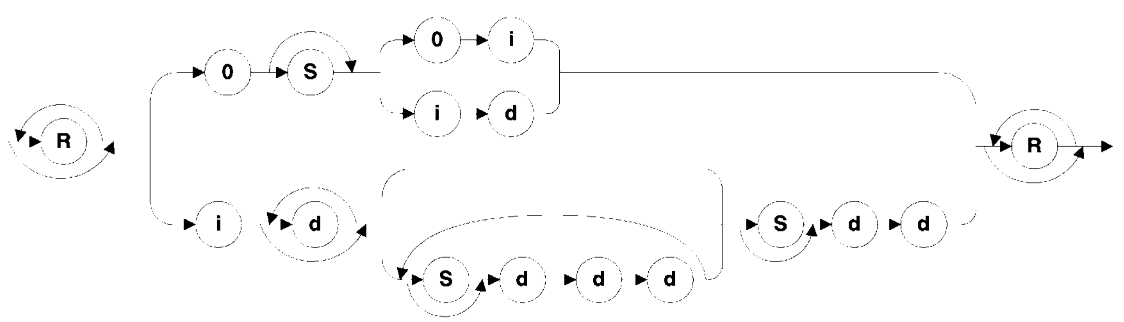


Fig. 13. Syntactic graph used to carry out the syntactic analysis, where N stands for noise such as "#," 0 for the digit "0," S for separators "." and ",", i for the digits ranging from 1 to 9, and d for the digits ranging from 0 to 9.

TABLE 2
The Original List of Hypotheses and the Same List after Postprocessing

(a)Original list of hypotheses		(b)List after post-processing	
Hypothesis	Probability	Hypothesis	Probability
1#0,00	0.95	140.00	0.892
140,00	0.90	170.00	0.792
170,00	0.80	1#0.00	0.009

It is worthy of remark that we just choose a value near 1 (0.99 in this case, but it could be 1) to confirm the hypotheses verified by the automaton and a value near to 0 to penalize those not verified. We just do not use 0 because we opted to keep all hypotheses generated by the system in the final list. An efficient and very often used strategy for postprocessing is to estimate $P(M, V)$ from some data. For example, Lethelier et al. in [23] use an ergotic HMM model to express $P(M, V)$. It was possible because they had a real and huge database to train such a model. In our case, we are working with a laboratory database where the probability of occurrence of all amounts is the same. For example, the fact that very small amounts and very large amounts are less likely on checks does not happen in our database. Moreover, we are dealing with a small database (about 1,300 images in the training set) and, consequently, it is impossible to model all variability of the numerical amount from such a small data.

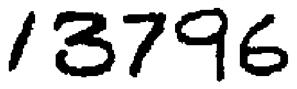
4.6 Global Decision

The global decision module decides either to accept the recognition result or reject it. The goal of rejection is to minimize the number of recognition errors for a given number of rejects. A direct scheme of rejection is to reject the image that has a global probability less than a determined threshold. However, due to the probabilistic model used, a 10-digit string usually supplies a global probability (P_{global}) smaller than a 2-digit string. Among the different strategies that we have tested, the one proposed by Fumera et al. [11] provided the better error-reject tradeoff for our system.

Basically, this technique suggests the use of multiple reject thresholds for the different data classes ($T_0, \ldots, T_n$) to obtain the optimal decision and reject regions. In order to define such thresholds, we have developed an iterative algorithm which takes into account a decreasing function of the thresholds variables $R(T_0, \ldots, T_n)$ and a fixed error rate T_{error}. We start from all threshold values equal to 1, i.e., the error rate equal to 0 since all images are rejected. Then, at each step, the algorithm decreases the value of one of the thresholds in order to increase the accuracy until the error rate exceeds T_{error}. The error rate is defined in (14).

Thereafter, the rejection of an image is straightforward. We just compare the probability of the components, which are recovered by backtracking the best path produced by the Viterbi algorithm, with their correspondent threshold. If any of the components has the probability less than its correspondent threshold, the entire string is rejected, otherwise, it is accepted.

Fig. 14 exemplifies the global decision. In such a case, the string will be accepted if $P(1) > T_1$, $P(3) > T_3$, etc. We will see in the next section that this strategy of rejection produces interesting error-reject tradeoffs. We will present experiments considering different T_{error}.



(a)

P(1)	P(3)	P(7)	P(9)	P(6)
0.99	0.95	0.98	0.99	0.99

(b)

T_1	T_3	T_7	T_9	T_6

(c)

Fig. 14. Rejection mechanism: (a) Digit string, (b) probability of each component, and (c) thresholds for the different classes.

5 Experiments and Results

In order to validate the concept of modular system as well as to show the robustness of our system, we ran experiments on two databases. The first database is composed of 2,000 images of numerical amounts and it aims at evaluating the performance of the system on recognition of numerical amounts on Brazilian bank checks. The second database is the NIST SD19 (hsf_7 series) and it aims at validating the concept of modular system as well as to show the robustness of our system on a well-known database. For all reported results, we used the following definitions of the recognition rate, error rate, rejection rate, and reliability rate. Let B be a test set with N_B string images. If the recognition system rejects N_{rej}, classifies correctly N_{rec}, and misclassifies the remaining N_{err}, then

$$\text{Recognition Rate} = \frac{N_{rec}}{N_B} \times 100, \tag{13}$$

$$\text{Error Rate} = \frac{N_{err}}{N_B} \times 100, \tag{14}$$

$$\text{Rejection Rate} = \frac{N_{rej}}{N_B} \times 100, \tag{15}$$

$$\text{Reliability} = \frac{\text{Recognition Rate}}{\text{Recognition Rate} \quad + \quad \text{Error Rate}} \times 100. \tag{16}$$

Therefore, the recognition rate, error rate, and rejection rate sum up to 100 percent.

5.1 Experiments on Numerical Amounts

For our experiments a database containing 2,000 images of numerical amounts was used. Most images of this database have a nonnumerical symbol ("#") affixed at the beginning and at the end of the numerical amount, a period "." to delimit a 3-tuple of digits, and a comma "," to identify the cents portion in the numerical amount. This database was divided in the following way: 1,300, 200, and 500 images for training, validation, and testing, respectively. From these three databases, we extracted 11,400, 2,000, and 4,000 isolated characters for training, validation, and testing, respectively. For all three sets, 80 percent of the database consists of digit images while the rest is composed of nonnumerical images ("#," ",", and "."). Therefore, we have used the numeric part of the databases (80 percent) to train e_{10} and the rest (20 percent) to train e_3. The recognition rates achieved by e_{10}, e_3 on the test set were 99.2 percent, 99.0 percent, respectively (zero-rejection level). Thereafter, we submitted the training set to

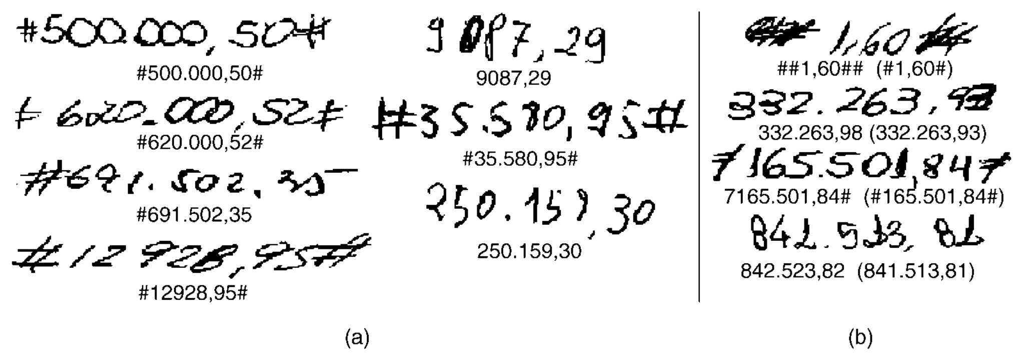


Fig. 15. Examples of numerical amounts on Brazilian bank checks: (a) Recognized and (b) not recognized (the correct string is the one in parentheses).

e_{10} and e_3 and trained the classifier e_{13} with the outputs provided by e_{10} and e_3 plus the five components described in Section 4.3.1. e_{13} achieved a recognition rate of 98.9 percent on the test set.

As discussed in Section 4.3.2, the verifier v_o has two outputs: isolated characters and oversegmentation. In order to train this verifier, we have used the following data: 8,000 correctly segmented characters, 8,000 naturally isolated characters, and 12,000 oversegmented parts which were generated automatically by the segmentation algorithm through the segmentation of the isolated and touching characters. The first two parts are devoted to train the first class of the verifier, while the third one is devoted to train the second class. Therefore, the training set used by v_o is composed of 28,000 samples. The validation and test sets were built in the same manner, and they have 14,000 samples each. This verifier reached a recognition rate of 99.40 percent on the test set.

Finally, we have built the database for v_u. As described in Section 4.3.3, the task of such a verifier is to detect the undersegmentation. Thus, it considers two classes: isolated characters and under-segmentation. The database used in this case is composed of 9,000 samples which are divided into 5,000 images of isolated characters and 4,000 images of touching characters. The validation and test sets were built considering the same distribution of samples and they are composed of 4,000 samples each. This verifier reached a recognition rate of 99.17 percent on the test set.

After training the classifier and verifiers, we carried out some experiments using the test set of numerical amounts (500 images). The number of characters per image in this database (average length) is about nine. Fig. 15 shows some examples of numerical amounts on Brazilian bank checks extracted from our database.

Table 3 shows the recognition rates (zero-rejection level) achieved in four different configurations of the system. A numerical amount image is counted as correctly classified if all characters composing it are correctly classified. This table allows us to evaluate all possible configurations of the recognition system and also verify the superiority of the configuration that considers all system modules. By analyzing the recognition rates of all system configurations, we can notice that the verifiers and the postprocessor are complementary in some respects, once the postprocessor resolved some problems where the verifiers failed and vice versa.

Basically, the postprocessor differentiates the digits from symbols, e.g., 140,00 confused with 1#0,00. In such a case, the automaton does not accept a symbol between two digits. Considering the last experiment (e_{13}, v_o, v_u, pp), we divided the total error of the system into three classes: segmentation, recognition, and verification. The segmentation errors are caused by undersegmentation which is due to a lack of basic points in the neighborhood to the connection stroke [27] (1.9 percent). The recognition errors are confusions of the general-purpose recognizer (51.7 percent), confusion generated by segmentation effects, such as ligatures and noises produced by segmentation cuts (30.8 percent), and confusions generated by fragmentation (7.6 percent). The latter occurs when the system misclassifies a broken character, which usually are due to natural fragmentation caused by the handwriting style (Fig. 18c) and noises acquired during scanning or preprocessing. (Fig. 18d).

In spite of the fact that the verifiers supplied a remarkable improvement in terms of recognition rates, they produced a new class of errors, which is related to the confusion between the isolated and the oversegmented characters (verifier v_o) and the confusion between the isolated and the undersegmented characters (verifier v_u) (13.1 percent). In the next section, we discuss in more detail this kind of errors.

Since bank check systems demand low error rates, we ran two experiments (based on the configuration "e_{13}, v_o, v_u, pp"), where we fixed the error rates at 0.5 and 0.1 percent, respectively. Table 4 presents recognition, rejection, and reliability rates at these two error levels while Fig. 16 shows the error-reject tradeoff for the same experiment.

TABLE 3
Recognition Rates (%) for the Numerical Amounts (Zero-Rejection Level)

System Modules			
e_{13}	e_{13}, v_o, v_u	e_{13}, pp	e_{13}, v_o, v_u, pp
34.35	71.51	47.91	77.49

TABLE 4
Recognition Rates (Rec.), Rejection Rates (Rej.), and Reliability Rates (Rel.) on the Numerical Amounts for Different Error Rates

Error 0.5%			Error 0.1%		
Rec.(%)	Rej.(%)	Rel.(%)	Rec.(%)	Rej.(%)	Rel.(%)
57.17	42.33	99.13	56.57	43.33	99.82

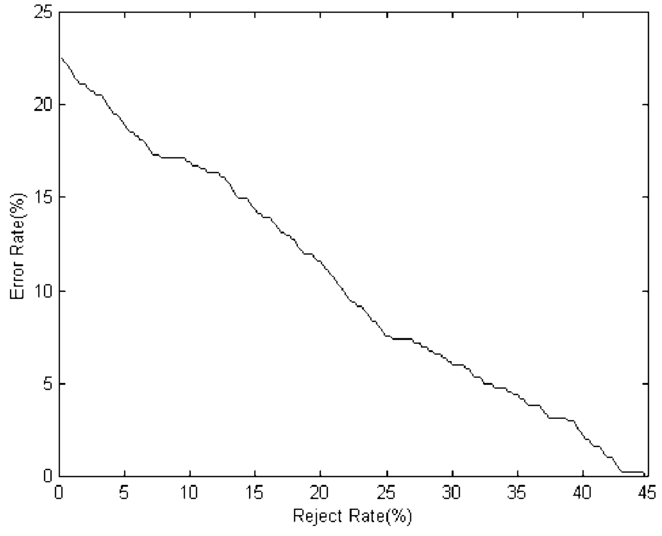


Fig. 16. Error rate versus rejection rate for numerical amounts.

5.2 Experiments on NIST SD19

The SD19 database, which is an update of SD3 and SD7 [12], is provided by the American National Institute of Standards and Technology (NIST). This database contains the full page binary images of 3,699 Handwriting Sample Forms (HSFs) and 814,255 segmented handprinted digit and alphabetic characters from those forms.

Since this database is composed of digit strings only, all Task Dependent Modules (gray boxes in Fig. 1) related to the numerical amount task were removed, except the Postprocessor module which was changed to verify the length of the string. Since we are dealing with different handwriting styles (Brazilian and North American), the general-purpose recognizer, in this case (e_{10}), oversegmentation verifier (v_o), and undersegmentation verifier (v_u) were retrained.

In order to train e_{10}, we have used the NIST SD19 in the following way: the training and validation sets were composed of 195,000 and 28,000 samples from hsf_{0, 1, 2, 3}, respectively, while the test set was composed of 60,089 samples from hsf_7. The recognition rates (zero-rejection level) achieved by e_{10} were 99.66 percent, 99.65 percent, and 99.13 percent on the training, validation, and test sets, respectively. The verifiers were trained using the same methodology and number of samples described in the previous section. The recognition rates achieved by the oversegmentation and undersegmentation verifiers were 99.41 percent and 99.25 percent, respectively. The experiments using numeral strings are based on 12,802 numeral strings extracted from the hsf_7 series and distributed into six classes: 2_digit (2,370), 3_digit (2,385) 4_digit (2,345), 5_digit (2,316), 6_digit (2,316), and 10_digit (1,217) strings, respectively. These data exhibit different problems such as touching and fragmentation and they have also been used as a test set by Britto Jr. et al. [5].

Table 5 summarizes the results for these experiments. In this table, we can see the performance at the zero-rejection level for four different versions of the system. These results aim at showing the importance and contribution of each module to the global system. For the second experiment, which does not consider the postprocessor module (number of digits) we present also the performance at three error levels: 2, 1, and 0.5 percent. By comparing the two first experiments, we can observe the efficiency of the proposed verifiers on the NIST database, as well as conclude that such verifiers clearly improve the performance of the system. Fig. 17a shows the behavior for strings of different lengths (from 2 to 10 digits), while Fig. 17b exhibits the error-reject tradeoff for all used fields in the test set.

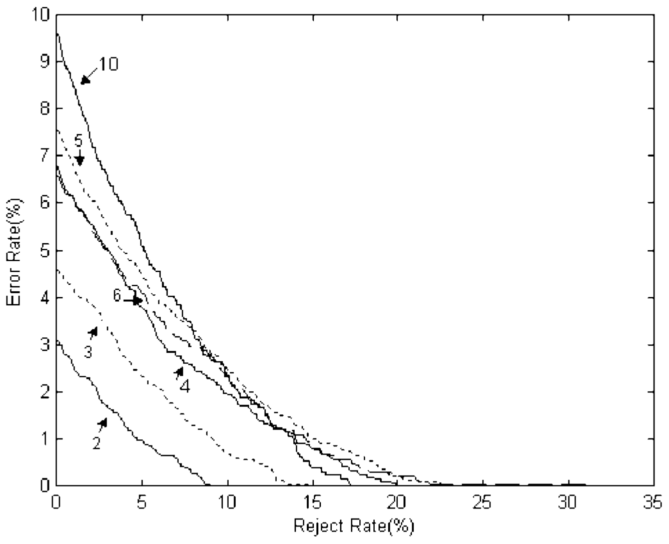
By analyzing the system errors, we observed that they can be classified into four classes: confusions generated by e_{10} (67 percent), errors caused by segmentation (9 percent), errors caused by fragmentation (11 percent), and confusions generated by the low-level verifiers v_o and v_u (12 percent). The first class, which is the most frequent one, is related to the weakness of the general-purpose recognizer but also the difficult cases found in the test database as shown in Fig. 18a. In this case, the digits 8 and 9 were classified as 6 and 7, respectively. The second class of errors is caused either by undersegmentation (20 percent) which is due to a lack of basic points in the neighborhood to the connection stroke or effects generated by the segmentation algorithm such as ligatures (80 percent) which can be visualized in Fig. 18b. The third class of errors is produced when the grouping algorithm fails. Usually, it happens to images with poor quality (Fig. 18d and sometimes images composed of two strokes (Fig. 18c)).

The last class of errors is produced by the low-level verifiers, where v_u is responsible for 87 percent and v_o for 13 percent of the verification errors. The most frequent confusions generated by v_u are related to specific configurations of the digits "6," "0," and "8" (Fig. 18e). In such cases, the digits have strong concavities and often they are damaged by preprocessing tools. Fig. 19a shows some examples of misclassified images, while Fig. 19b shows examples of images containing touching or broken characters that were correctly recognized by the system.

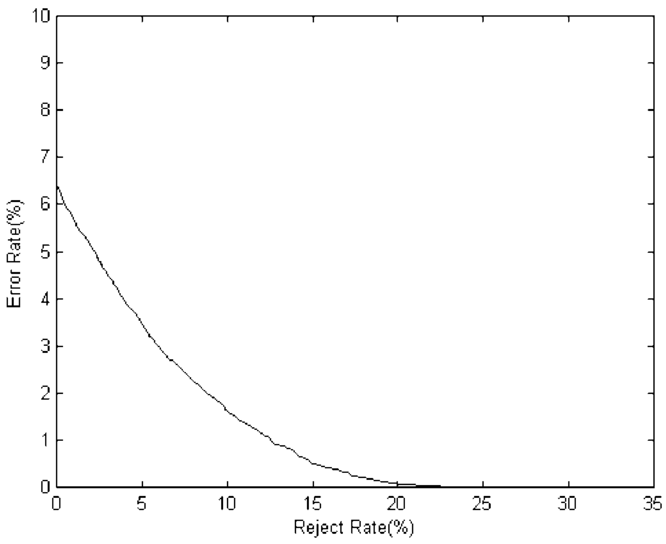
As we can notice, the part of the system that produces more errors is the general-purpose recognizer e_{10}. In order to reduce

TABLE 5
Recognition Rates for the NIST Experiment

String	Number of	System Modules - (zero-rejection level)				Error Rate (e_{10}, v_o, v_u)		
Length	Strings	e_{10}	e_{10}, v_o, v_u	e_{10}, pp	e_{10}, v_o, v_u, pp	2%	1%	0,5%
2	2370	91.56	**96.88**	96.41	97.21	96.08	94.93	93.88
3	2385	87.98	**95.38**	94.37	95.80	92.89	90.60	89.84
4	2345	84.91	**93.38**	91.96	94.11	88.18	85.78	84.36
5	2316	82.00	**92.40**	91.06	93.22	87.01	84.77	82.44
6	2169	85.66	**93.12**	94.12	95.90	88.16	85.68	84.03
10	1217	78.97	**90.24**	89.56	91.07	80.42	77.85	75.20



(a)



(b)

Fig. 17. Error rates versus rejection rates for the NIST database: (a) Error-reject tradeoff for strings of different lengths. (b) Error-reject tradeoff for all used fields in the test set.

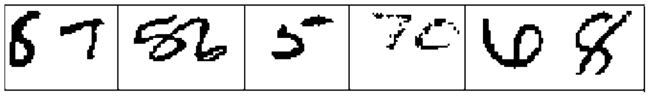
such errors and make the overall system more reliable, our next efforts will be focused on the optimization of the feature sets used by both the general-purpose recognizer and verifiers.

6 Discussion and Comparison

So far, we have described all system modules and how they interact with each other in order to make the entire system more reliable. We have also presented experiments on two different databases. We have shown that the low-level verifiers have brought remarkable improvements to the recognition system. We have seen that the modular framework proposed is suitable to process numerical fields in different applications. Thus, the results reported on numerical amounts and NIST database have been carried out using an identical system with the following exceptions: The optimization of the parameters for digit detection and grouping was carried out on different databases and the classifiers were trained on different databases as well.

Comparison with published methods is very delicate when we consider bank check recognition systems since different databases and formats are used, different nonnumerical classes are involved, and different sizes of databases are considered. For example, Lethelier et al. [23] present a system for French bank checks which copes with four nonnumerical classes ("-", ".", ",", "F"). They claim a recognition rate of 60 percent (zero-rejection level) on a test set of 10,000 images of French bank checks. Kaufmann and Bunke [18] propose a system for Swiss postal checks that neither considers nonnumerical classes nor the cents portion. The result achieved by this system is 79.3 percent (zero-rejection level) and 58.1 percent with an error rate of 0.2 percent.

Regarding the publications using the NIST database a more detailed comparison is possible. Table 6 summarizes the recognition rates at different error levels for the papers [5], [22], [14]. Ha et al. [14] used about 5,000 strings of the NIST SD3. Lee and Kim [22] used 5,000 strings but they did not specify the used data. Britto Jr. et al. [5] used the same database we are using, i.e., 12,802 strings of the NIST SD19 (hsf_7 series), however, they used the knowledge about the number of numerals composing the string. In Table 6, the "-" indicates that no recognition rate has been reported for the specified error rate. By comparing the results reached by our system (Table 5) with those reported by other authors (Table 6), we can confirm that our system provides very good recognition rates at zero-recognition level and a very encouraging error-reject tradeoff.



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

Fig. 18. Different classes of errors generated by the system: (a) Recognition errors, (b) ligature between two digits, (c) natural fragmentation, (d) fragmentation caused by noise, and (e) isolated digits classified as undersegmentation by v_u.

7 Conclusion

We have proposed a modular offline system that can cope with different applications. Our main focus has been the recognition of numerical amounts on Brazilian bank checks and numeral strings of the NIST SD19 (hsf_7 series). Our system takes a segmentation-based recognition approach where an explicit segmentation is employed. Combination of different levels such as segmentation, recognition, and postprocessing is made within a multihypothesis approach and a probabilistic model which allows a sound integration of all knowledge sources used to infer a plausible interpretation.

We have shown a very efficient scheme of verification to deal with oversegmentation and undersegmentation problems. Such a scheme takes into account two low-level verifiers. The first verifier uses a new feature set which is based on multilevel concavity analysis and contextual

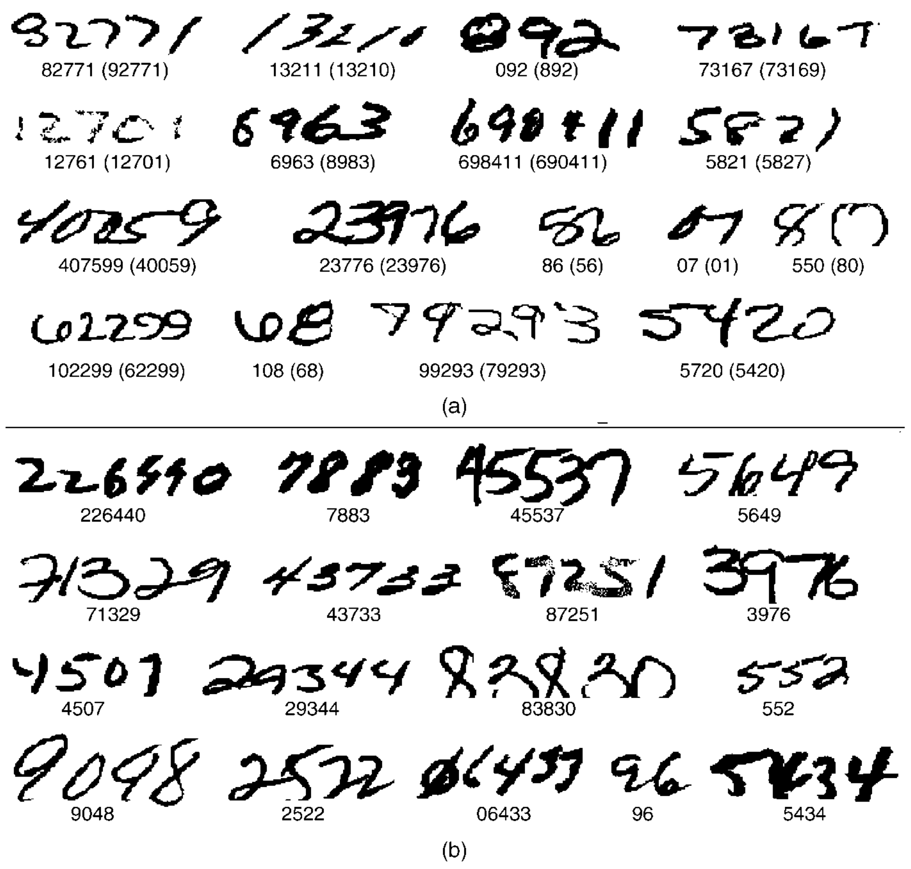


Fig. 19. Examples of digit strings (NIST SD19): (a) Not recognized (the correct string is the one in parenthesis) and (b) recognized.

information, in order to reduce the confusion between isolated and oversegmented characters. The second one works on the opposite problem, i.e., eliminating the confusion between isolated and undersegmented characters.

In order to improve the overall performance of the system that deals with numerical amounts on Brazilian bank checks, we have developed a simple and efficient postprocessor which is based on a deterministic automaton. This module provided an improvement of about 6 percent in the recognition rate. Finally, the rejection mechanism employed minimizes the number of rejection errors for a given number of rejects.

Comprehensive experiments on numerical amounts and NIST SD19 databases have been conducted. High recognition

TABLE 6
Recognition Rates on NIST Databases Reported by Other Authors

Authors	String Length	Number of tested strings	Error rate 0%	Error Rate 2%	1%	0,5%
Ref. [14]	2	981	96.20	94.50	93.50	91.50
	3	986	92.70	86.00	79.50	70.50
	4	988	93.20	86.50	81.00	70.00
	5	988	91.10	81.00	77.50	70.50
	6	982	90.30	80.50	75.50	66.50
Ref. [22]	2	1000	95.23	-	95.20	-
	3	1000	88.01	-	87.90	-
	4	1000	80.69	-	80.50	-
	5	1000	78.61	-	78.40	-
	6	1000	70.49	-	70.20	-
Ref. [5]	2	2370	95.23	-	-	-
	3	2385	92.62	-	-	-
	4	2345	92.11	-	-	-
	5	2316	90.00	-	-	-
	6	2169	90.09	-	-	-
	10	1217	86.94	-	-	-

rates at zero-rejection level and a very encouraging error-reject tradeoff have been obtained. The results reached by our system compare favorably to other published methods.

ACKNOWLEDGMENTS

This work was supported by Fundação Araucária, CENPARMI, and NSERC of Canada. The authors would like to thank A. El-Yacoubi and E. Lethelier for their important contribution in this work and also the anonymous reviewers for their helpful comments in improving the earlier draft of this paper.

REFERENCES

[1] Banco Central do Brasil, *Manual de Normas e Instruções—Circular nº 001825,* Oct. 1990. http://www.bcb.gov.br/mPag.asp?codP=106&cod=112&perfil=1 (in portuguese).
[2] C.M. Bishop, *Neural Networks for Pattern Recognition.* Oxford, U.K.: Oxford Univ. Press, 1995.
[3] H. Bourlard and N. Morgan, "Merging Multilayer Perceptrons and Hidden Markov Models: Some Experiments in Continuous Speech Recognition," *Neural Networks: Advances and Applications,* E. Gelenbe, ed., North Holland Press, 1991.
[4] J.S. Bridle, "Training Stochastic Model Recognition Algorithms as Networks can Lead to Maximum Mutual Information Estimation of Parameters," *Proc. Advances in Neural Information Processing Systems,* vol. 2, pp. 211-217 1990.
[5] A. Britto Jr., R. Sabourin, F. Bortolozzi, and C.Y. Suen, "A Two-Stage HMM-Based System for Recognizing Handwritten Numeral Strings," *Proc. Sixth Int'l Conf. Document Analysis and Recognition,* pp. 396-400, 2001.
[6] T.M. Bruel, "A System for the Off-Line Recognition of Handwritten Text," *Proc. 12th Int'l Conf. Pattern Recognition,* vol. 2, pp. 129-133, 1994.
[7] Y.K. Chen and J.F. Wang, "Segmentation of Single- or Multiple-Touching Handwritten Numeral String Using Background and Foreground Analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 11, pp. 1304-1317, Nov. 2000.
[8] S.J. Cho, J. Kim, and J.H. Kim, "Verification of Graphemes Using Neural Networks in an HMM-Based On-Line Korean Handwriting Recognition System," *Proc. Seventh Int'l Workshop Frontiers in Handwriting Recognition,* pp. 219-228, 2000.
[9] G. Dimauro, S. Impedovo, G. Pirlo, and A. Salzo, "Automatic Bankcheck Processing: A New Engineered System," *Int'l J. Pattern Recognition and Artificial Intelligence,* S. Impedovo et al., eds., pp. 467-503, 1997.
[10] G.D. Forney Jr., "The Viterbi Algorithm," *Proc. IEEE,* vol. 61, no. 3, pp. 268-278, 1973.
[11] G. Fumera, F. Roli, and G. Giacinto, "Reject Option with Multiple Thresholds," *Pattern Recognition,* vol. 33, no. 12, pp. 2099-2101, 2000.
[12] P.J. Grother, "NIST Special Database 19-Handprinted Forms and Characters Database," Nat'l Inst. Standards and Technology (NIST), 1995.
[13] T.M. Ha and H. Bunke, "Off-Line, Handwritten Numeral Recognition by Perturbation Method," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 5, pp. 535-539, May 1997.
[14] T.M. Ha, M. Zimmermann, and H. Bunke, "Off-Line Handwritten Numeral String Recognition by Combining Segmentation-Based and Segmentation-Free Methods," *Pattern Recognition,* vol. 31, no. 3, pp. 257-272, 1998.
[15] L. Heutte, J. Moreau, B. Plessis, J. Plagmaud, and Y. Lecourtier, "Handwritten Numeral Recognition Based on Multiple Feature Extractors," *Proc. Int'l Conf. Document Analysis and Recognition,* pp. 167-170, 1993.
[16] L. Heutte, P. Pereira, O. Bougeois, J. Moreau, B. Plessis, and P. Courtellemont, "Multi-Bank Check Recognition System: Consideration on the Numeral Amount Recognition Module," *Int'l J. Pattern Recognition and Artificial Intelligence,* S. Impedovo et al., eds., pp. 595-617, 1997.
[17] J. Cai and Z.Q. Liu, "Integration of Structural and Statistical Information for Unconstrained Handwritten Numeral Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 3, pp. 263-270 Mar. 1999.
[18] G. Kaufmann and H. Bunke, "Automated Reading of Cheque Amounts," *Pattern Analysis and Applications,* vol. 3, no. 2, pp. 132-141, 2000.
[19] S. Knerr, V. Anisimov, O. Baret, N. Gorski, D. Price, and J. Simon, "The A2iA Intercheque System: Courtesy Amount and Legal Amount Recognition for French Checks," *Int'l J. Pattern Recognition and Artificial Intelligence,* S. Impedovo et al., eds., pp. 505-547, 1997.
[20] G.E. Kopec and P.A. Chou, "Document Image Decoding Using Markov Source Models," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 16, no. 6, pp. 602-617 June 1994.
[21] Y. LeCun, B. Boser, J.S. Denker, B. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel, "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation,* vol. 1, no. 4, pp. 541-551, 1989.
[22] S.W. Lee and S.Y. Kim, "Integrated Segmentation and Recognition of Handwritten Numerals with Cascade Neural Networks," *IEEE Trans. Systems, Man, and Cybernetics, Part C: Applications and Reviews,* vol. 29, no. 2, pp. 285-290, 1999.
[23] E. Lethelier, M. Leroux, and M. Gilloux, "An Automatic Reading System for Handwritten Numeral Amounts on French Checks," *Proc. Third Int'l Conf. Document Analysis and Recognition,* pp. 92-97, 1995.
[24] H.C. Leung, "Speech Recognition Using Stochastic Explicit-Segment Moduling," *Proc. Second European Conf. Speech Communication and Technology,* pp. 931-934 1991.
[25] R. P. Lippmann, "Pattern Classification Using Neural Networks," *IEEE Comm. Magazine,* vol. 27, no. 11, pp. 47-64, 1989.
[26] O. Matan and C. Burges, "Recognizing Overlapping Hand-Printed Characters by Centered-Objects Integrated Segmentation and Recognition," *Proc. Int'l Joint Conf. Neural Networks,* pp. 504-511, 1991.
[27] L.S. Oliveira, E. Lethelier, F. Bortolozzi, and R. Sabourin, "A New Segmentation Approach for Handwritten Digits," *Proc. 15th Int'l Conf. Pattern Recognition,* vol. 2, pp. 323-326, 2000.
[28] L.S. Oliveira, R. Sabourin, F. Bortolozzi, and C.Y. Suen, "A Modular System to Recognize Numerical Amounts on Brazilian Bank Cheques," *Proc. Sixth Int'l Conf. Document Analysis and Recognition,* pp. 389-394, 2001.
[29] M.D. Richard and R.P. Lippmann, "Neural Network Classifiers Estimate Bayesian a Posteriori Probabilities," *Neural Computation,* vol. 3, no. 4, pp. 461-483, 1991.
[30] M. Shridhar and A. Badreldin, "Recognition of Isolated and Simply Connected Handwritten Numerals," *Pattern Recognition,* vol. 19, no. 1, pp. 1-12, 1986.
[31] H. Takahashi and T. Griffin, "Recogniton Enhancement by Linear Tournament Verification," *Proc. Second Int'l Conf. Document Analysis and Recognition,* pp. 585-588, 1993.
[32] J. Zhou, Q. Gan, A. Krzyzak, and C.Y. Suen, "Recognition and Verification of Touching Handwritten Numerals," *Proc. Seventh Int'l Workshop Fontiers in Handwriting Recognition,* pp. 179-188, 2000.

Luiz S. Oliveira received the BS degree in computer science from UnicenP, Curitiba, PR, Brazil and the MSc degree in electrical engineering and industrial informatics from the Centro Federal de Educação Tecnológica do Paraná (CEFET-PR), Curitiba, PR, Brazil, in 1995 and 1998, respectively. From 1994 to 1998, he was a system analyst at HSBC Bank, Curitiba, PR, Brazil, where he worked on financial systems. Currently, he is a PhD candidate at École de Technologie Supérieure, Université du Québec, Montréal, Canada and a visiting scientist at the Centre for Pattern Recognition and Machine Intelligence (CENPARMI). His current interests include pattern recognition, neural networks, image analysis, and evolutionary computation.

Robert Sabourin received the Bing, MscA, and PhD 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 Pontifícia Universidade Católica do Paraná (PUCPR, Curitiba, Brazil) where he has been coresponsible since 1998 for the implementation of a PhD program in applied informatics. Since 1996, he has been 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. He is a member of the IEEE.

Flávio Bortolozzi received the BS degree in mathematics in 1977 from Pontifícia Universidade Católica do Paraná (PUC-PR), Brazil, the BS degree in civil engineering in 1980 from PUC-PR, and the PhD degree in system engineering (computer vision) from the Université de Technologie de Compiègne, France, in 1990, where his work was concerned with 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 in the Computer Science Department and the prorector for research at PUC-PR. His research interests are computer vision, handwriting recognition, document image analysis, educational multimedia, and hypermedia.

Ching Y. Suen received the MSc (Eng.) degree from the University of Hong Kong and the PhD 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. Professor 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. He is 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 the International Conference on Document Analysis and Recognition, the International Workshop on the Fontiers of Handwriting Recognition, 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.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** http://computer.org/publications/dlib.