Off-line Handwritten Numeral String Recognition by Combining Segmentation-based and Segmentation-free Methods

Thien M. HA, Matthias ZIMMERMANN, and Horst BUNKE

University of Berne

Institut für Informatik und Angewandte Mathematik

Neuebrückstr. 10, CH-3012 Berne, Switzerland

E-Mail: haminh@iam.unibe.ch

Abstract

We propose an off-line system for the recognition of handwritten numeral strings. The system is built upon a number of components for presegmentation, digit detection, isolated digit recognition, segmentation-free recognition of numeral strings and global decision making. Presegmentation consists in dividing the input numeral string image into groups of numerals, each of which represents an arbitrary number of numerals. For each group, the actual number of numerals and their identity are determined by a cascade of two recognition methods: the first is for isolated numerals, while the second is a segmentation-free method that is able to recognize a numeral group of any length. In the global decision module all results from both the digit detection and the segmentation-free module are merged together and an accept/reject decision is made. We also introduce the concept of dummy symbol in order to overcome the problem of noisy parts that cannot be eliminated by standard filtering algorithms. Comprehensive experiments on data from both the CEDAR and NIST SD3 database have been carried out. Recognition rates of 83.6%

(CEDAR) and 92.7% (NIST) at the string level were achieved. These results compare favorably to other published methods.

Key Words: Handwritten Numeral String, Segmentation-Based Recognition, Segmentation-Free Recognition, Isolated Numeral Recognition, Dummy Symbol, Accept/Reject Decision.

1 Introduction

Automatic character recognition is a subfield of pattern recognition and can either be on-line or off-line. On-line recognition refers to those systems where the data to be recognized is input through a tablet digitizer, which acquires the position of the pen tip as the user writes in real-time. In contrast, off-line systems input the data from a document through an acquisition device, such as a scanner or a camera. Off-line character recognition is moreover divided into two categories, namely, machine-printed and handwritten. This paper is concerned with one particular problem in off-line handwriting recognition, namely, the recognition of sequences of digits where individual digits may touch or overlap each other. Such sequences are usually produced in unconstrained, i.e. 'natural', handwriting.

Off-line handwritten numeral string recognition has been a topic of intensive research in recent years due to its large number of potential applications. Indeed, it is present in almost every application involving handwriting recognition, for instance, postal address, check, and tax form reading. In such environments, a numeral string can be written in printed, cursive, or mixed style. The recognition of numeral strings differs from that of isolated numerals because it requires the segmentation of a string into separate entities representing individual numerals. It is also different from the problem of recognizing cursively handwritten words from a dictionary in the sense that virtually no context is available, i.e., any numeral can follow any other one. Segmentation of numeral string is generally a difficult task because individual numerals in a string can overlap or touch each other, or a numeral can be broken into several parts. For example the horizontal bar of the numeral 5 may be separated from the rest. Many methods have been proposed to tackle these problems. They can be classified into two approaches, namely, discrete and continuous. See [3] for a recent survey.

In the discrete approach, which includes both segmentation-based and segmentation-free methods, the numeral isolation (separation) takes place at a number of points where the image exhibits some special characteristics. For instance, the analysis of the vertical projection of black pixels provides a simple (but not always correct or sufficient) way to segment the input string into numerals or groups of numerals. Other special characteristics are line end-points, crossings, and T-joints [21]. For segmentation-based methods, these special points are carefully chosen so that they provide correct cuttings of the input image without wrongly splitting any numeral across its body. This task is difficult and has been shown to be unreliable [6]. The segmentation-free methods remedy these problems [21], [6]. Each split is validated by submitting its left- and right-part to an isolated recognition system. The resulting confidences are then used as a measurement of the likelihood of the split being correct. Thus, all potential splitting points are examined and the system chooses the subset to be the final result that maximizes a global likelihood. Usually, the maximization is achieved by dynamic programming [8], a statistical formalism [9], or graph search [21], [10]. See also [11], [12] for other implementation versions.

In the continuous approach, a sliding window scans the input image from left to right. Each position of the window defines a sub-image which is extracted and analyzed. Since it is generally impossible to have a window width that fits all individual numerals in a string, some mechanism must be provided to handle this variability. For instance, the window width is chosen to be roughly two times the average width of the individual numerals so as to be almost certain that it covers any numeral in a string at least once through the scanning [13]. Another method consists in using a recurrent neural network (a general architecture that includes time-delay neural networks) to encode information between neighboring vertical strips (thin vertical sliding windows) [14]. Other similar mechanisms have also been used [15], [16].

At present time it is not clear which approach, discrete or continuous, is better. Interestingly enough, most discrete methods report their results using the CEDAR database [25] whereas the continuous methods tend to use the NIST database [29] (see Figs. 1 and 2). The former contains *totally unconstrained* data collected from live mail whereas the latter contains *slightly constrained* data (the writers were requested to write the numeral strings in preprinted boxes). There seems to be a strong correlation between data and methods.

Figure 1: Numeral String Samples from CEDAR Database

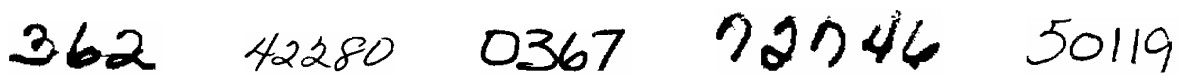


Figure 2: Numeral String Samples from NIST SD3 Database

The system presented in this paper is based on the discrete approach, but we will present our results on both the CEDAR and NIST databases, thus providing a first cross-validation. The system architecture appropriately combines segmentation-based and segmentation-free methods. This combination takes advantages of one method to compensate for the drawbacks of the other, and vice versa.

Section 2 gives an overview of the recognition system, Sections 3 to 6 describe its modules, experiments are reported in Section 7, discussion and comparisons is presented in Section 8 and the paper is concluded in Section 9.

2 System Overview

The simplest methods in the discrete approach are segmentation-based, where first the string is segmented into entities susceptible to represent individual numerals, and then each entity is fed into an isolated numeral recognizer. Several segmentation methods have been proposed, ranging from the simple connected component analysis to sophisticated schemes combining different techniques. For instance, [6] uses upper/lower contours and heuristics to perform segmentation and [1] makes use of both contour and skeleton. See also [7], [20] for other methods. All these methods may be characterized as 'blind' segmentation, since they do not explicitly take into account the particular set of underlying symbols to determine potential break points.

More sophisticated discrete methods are segmentation-free in that candidate breaks are cho-

sen at the elementary stroke level, and each candidate is validated by an isolated numeral recognizer[21], [22]. In other words, each candidate breaking point is checked to see whether it is meaningful with respect to the set of underlying symbols (numerals) when combined with neighboring strokes. Therefore the choice is more rational than in the segmentation-free methods where heuristics prevail, at the price of a higher computational complexity, however. Another important drawback of the segmentation-free methods is that the segmentation error rate increases very fast with the number of numerals composing the whole string [21].

The method adopted in this paper belongs to a third category where both segmentation-based and segmentation-free methods are used [23], [24]. Similarities and differences w.r.t. previous works will be cited at appropriate places. The basic idea of our method consists in alleviating the importance of heuristics in the segmentation-based methods by means of a segmentation-free method while avoiding the main drawbacks of the latter, namely, its fast-dropping segmentation rate and its excessive computational burden for long numeral strings. More precisely, our system is composed of a cascade of two methods, the first – called presegmentation in the following – is segmentation-based, whereas the second is segmentation-free. In contrast with the usual segmentation-based methods where the numeral string is segmented into individual numerals, our presegmentation *only* attempts to segment it into groups of numerals. Each group should represent an arbitrary integer number of numerals, i.e. one, two, three, ..., numerals. All groups of numerals (with unknown lengths) are then separately recognized and the results are eventually merged together yielding the final interpretation; see Fig. 3. To avoid the two aforementioned drawbacks of segmentation-free methods, each output of presegmentation (a partial image) is first submitted to a 'Digit Detection' module and only if the partial image is rejected do we go to the 'Segmentation-free' module.

In real environments, the image of a numeral string usually contains many 'noisy' parts in addition to the numerals themselves. As an example Fig. 1 shows some strings with and without noisy parts [25]. To cope with these noisy parts, we introduce the concept of *dummy* symbol. That is, noisy parts that cannot be eliminated by standard filtering algorithms will be considered as dummy symbols. In our system, the treatment of dummy symbols is embedded in the 'Segmentation-free' module.

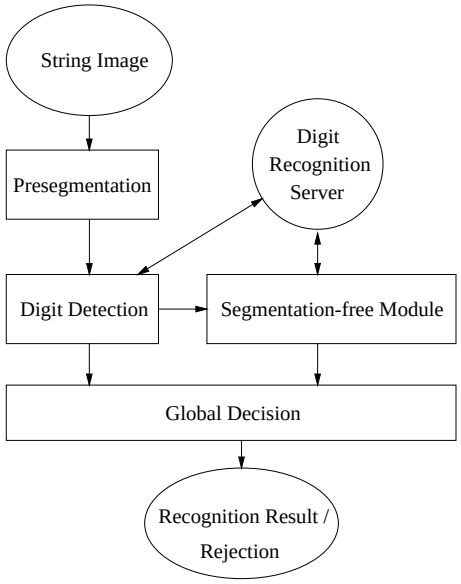


Figure 3: System Architecture.

The 'Presegmentation' module is described in Section 3, the 'Digit Detection' module in Section 4, the 'Segmentation-free' module in Section 5 and the 'Global Decision' module in Section 6. As 'Digit Recognition' server, we will use a method previously developed at our institute [18].

3 Presegmentation

The goal of 'Presegmentation' module is to divide the input numeral string, called *string image* (SI), into groups of numerals called *partial images* (PI's), each of which should represents an integer number of numerals. This allows us to convert the recognition of a SI to that of its PI's, thus reducing the complexity of subsequent operations (see Fig. 3). The presegmentation process operates in two steps, namely, connected component analysis and grouping. The connected component analysis segments the SI into *connected components* (CC) and eliminates very small CC's by filtering.

A CC can either represent an integer number of numerals or not. It is clear that the first situation is ideal because we assume that the 'Digit Detection' module or the 'Segmentation-free' module is able to handle it. The second situation where, for example, a CC is a broken part of

a numeral, is critical and should be avoided. This is achieved by first detecting potential broken parts and then grouping each of them to its neighbor(s).

The detection of broken parts is motivated by the following observation [1]. Among the ten classes of Arabic numerals, only two of them, namely, '4' and '5', are normally written in two strokes (see Fig. 4). The remaining eight classes are usually produced by one single stroke. Therefore the *a priori* probability of having broken parts in these two classes is higher than in the remaining eight classes. This leads to the following detection rule. A CC is regarded as a broken part if one of the following three conditions is met.

1. The CC does not intersect the median line (SI_{median}) of the numeral string.

2. $\frac{max(CC_{above},CC_{below})}{min(CC_{above},CC_{below})} > T_{broken1}; T_{broken1} \in [3,5]$ [2]
   where CC_{above} and CC_{below} denote the vertical height of the part above and below the median line respectively; see Fig. 4..

3. $\frac{CC_{height}}{SI_{height}} < T_{broken2}; T_{broken2} \in [0.2, 0.5]$.

Even when a CC intersects the median line, it may still be considered as broken part by the second condition if the intersection point is near to the top or bottom part of the CC. The third condition prevents too small CC's to be considered as isolated numerals. Although the above conditions are inspired from the numerals '4' and '5', it can be easily seen that they also cover those cases where a CC is not well aligned vertically with the whole string, thus becoming more likely to be a broken part.

In the grouping phase the CC's which are deemed broken parts are grouped to their neighborhood. We have to decide to which neighboring CC(s) a broken part CC_{broken} should be grouped. The decision is based on the following rule (see Fig. 4 for the definitions of various quantities):

[1] We consider only the North American writing style

[2] The intervals correspond to the examined search spaces during the optimization phase of the system

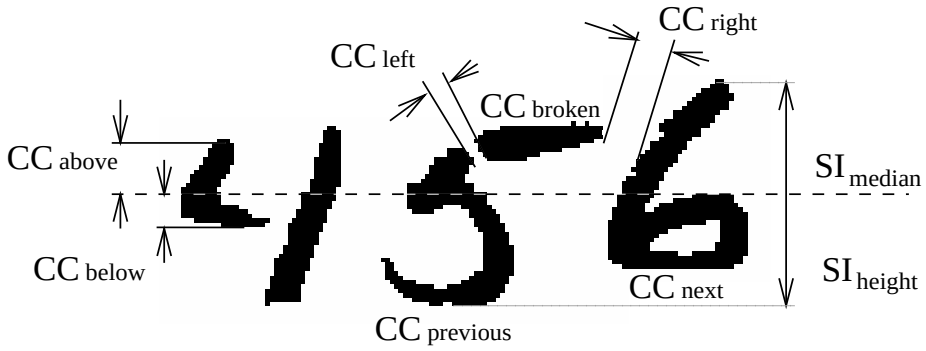


Figure 4: Definition of Various Geometric Quantities.

IF $(\frac{|CC_{left} - CC_{right}|}{SI_{height}} < T_{grouping})$ THEN

Group$(CC_{previous}, CC_{broken}, CC_{next})$

ELSE IF $(CC_{left} < CC_{right})$ THEN

Group$(CC_{previous}, CC_{broken})$

ELSE

Group(CC_{broken}, CC_{next})

where $T_{grouping} \in [0.2, 0.4]$. That is, when CC_{left} and CC_{right} are nearly equal, we prefer grouping the broken CC to both its left CC and right CC. The ambiguity will be resolved later by the 'Segmentation-free' module. If a broken CC is on the left or right end of the numeral string, CC_{left} or CC_{right} is set to a very high value to achieve the correct grouping. The grouping process is repeated as long as some groups can still be classified as broken parts. It can be seen that although the segmentation problem is not solved in this stage, we have avoided making some irremediable wrong decision(s). (A similar idea for the detection of broken parts has been proposed in [23]. The main difference resides in the grouping mechanism). The resulting PI's are ordered from left to right according their horizontal position in the SI. In the case of Fig. 4, there could be two resulting PI's corresponding to '4' and '56' respectively.

4 Digit Detection

For each PI resulting from the 'Presegmentation' module the 'Digit Detection' decides whether the PI contains an isolated digit or a group of digits. PI's containing isolated digits are recognized directly by the 'Digit Detection' module. PI's containing groups of numerals are rejected by the 'Digit Detection' and sent to the 'Segmentation-free' module for recognition. The decision to accept or reject a PI is based on a score value $S_{digit}(PI)$ which is computed separately for each PI. The score value is computed by combining recognition result from the 'Digit Recognition' server and additional "recognition" result based on the mean number of horizontal transitions from white to black over all scan-lines of a PI. We first motivate our approach and then explain the details.

Let us naively consider the problem from the pattern classification point of view, i.e., classify a PI as a digit or a group of more than one digits. It is a two class problem that can be, at least in principle, optimally solved by a standard classification technique [19]. Unfortunately, such an approach would be optimal only locally, i.e. for 'Digit Detection', but global optimality of the whole system would not be guaranteed. Moreover, even an attempt to provide only local optimality would require extremely complex analyses to determine the class *à priori* probabilities and the loss function. For instance, the loss of classifying a group of digits as one digit is different from the contrary, because the latter is not a real error in the sense that a PI containing one digit can still be correctly recognized by the 'Segmentation-free' module. Therefore, we decided to concentrate our efforts on another aspect of the problem. That is, the kind of measures or features that make one digit different from many digits. After some trials, it turned out that the number of horizontal transitions provide a good basic feature. To render it more reliable, we use the mean value over all scan-lines. Moreover the feature is made class-specific in combination with the result from 'Digit Recognition'.

To compute the score value $S_{digit}(PI) \in [0,1]$ the posterior probabilities $q_k(PI)$ from the 'Digit Recognition' server and the transition qualities $t_k(PI)$ are combined in the following way.

$$S_{digit}(PI) = \frac{1}{10} \sum_{k=0}^{9} q_k(PI) \times t_k(PI) \tag{1}$$

The computation of $q_k(PI)$ and $t_k(PI)$ will be explained in the following two paragraphs. The

reason for which we sum up over all ten numeral classes is that we are mainly interested in measuring how valid a PI is rather than how good it is with respect to the best class (a valid PI may belong to any of the ten numeral classes). This is in total contrast with other previous works where S_{digit} was defined as the maximum score (or likelihood) over all classes [24], [21].

The 'Digit Recognition' server simply receives an input image and outputs its class scores (posterior probabilities) $q_k \in [0,1], k = 0,\ldots,9$. The recognition method has been developed earlier at our institute [18] and uses a combination of two statistical sub-recognizers. The first sub-recognizers uses projection-based features whereas the second computes its features from contour information. For both sub-recognizers a fully connected feed-forward multi-layer perceptron is used for classification.

The mean number of horizontal transitions $\hat{\mu}_k$ for each digit class k and its standard deviation $\hat{\sigma}_k$ are estimated from measurements over some hundreds of isolated digits in the training phase. In the recognition phase for a given PI the mean number of horizontal transitions over all scanlines PI_{mt} is computed and used to estimate the transition quality $t_k(PI) \in [0,1], k = 0,\ldots,9$, of the PI regarding class k by the following equation.

$$t_k(PI) = 1 - erf(\frac{|PI_{mt} - \hat{\mu}_k|}{\hat{\sigma}_k}) \tag{2}$$

where $erf(.)$ is the standard *error function* ($erf(0) = 0, erf(\infty) = 1, erf(-x) = -erf(x)$). Similar to q_k, a high value for t_k indicates a high "recognition" confidence. In a sense, $t_k(PI)$ can be considered as the output of an isolated numeral recognizer with only one feature. This recognizer is not able to distinguish between, say, a '6' and a '9', because both classes have similar mean and standard deviation. But it provides a very good discrimination between a valid and an invalid PI, especially when the invalid PI is composed of more than one numeral.

The decision whether the PI represents an isolated digit (or a group of digits) is made by the following rule.

IF $(S_{digit}(PI) > T_{digit})$ THEN

Accept PI, send result to

the 'Global Decision' module

ELSE

Reject PI, send PI to

the 'Segmentation-free' module

where $T_{digit} \in [0.015, 0.025]$ is a predefined threshold found in the optimization phase. If the PI is accepted as an isolated digit, the class k which satisfies the following equation

$$q_k(PI) = \max_{k'} q_{k'}(PI) \tag{3}$$

is sent together with the corresponding value $q_k(PI)$ to the 'Global Decision' module. The class k is used to build up the global recognition result and the value $q_k(PI)$ is used for the global accept/reject decision.

5 Segmentation-free Module

The 'Segmentation-free' module is designed to recognize PI's containing groups of broken and/or touching digits. It consists of two levels, namely, 'Local Level' and 'Global Level'. The 'Local Level' is responsible for dividing a PI into *partial shapes* (PS) representing single digits and recognizing them. The 'Global Level' chooses the best among all possible combinations of PS's. An overview of the 'Segmentation-free' method is shown in Fig. 5.

Subsection 5.1 describes the different steps used in 'Local Level' processing. The 'Global Level' processing is explained in Subsection 5.2. A special feature of the 'Segmentation-free' module is the concept of *dummy symbol*. It is presented in Subsection 5.3 and allows to treat noisy parts that cannot be eliminated by standard filtering algorithms.

5.1 Local Level Processing

At the 'Local Level' processing the PI is decomposed into PS's that could represent single digits. The PS's have to meet simple size and shape constraints to be accepted. See Fig. 8 for an

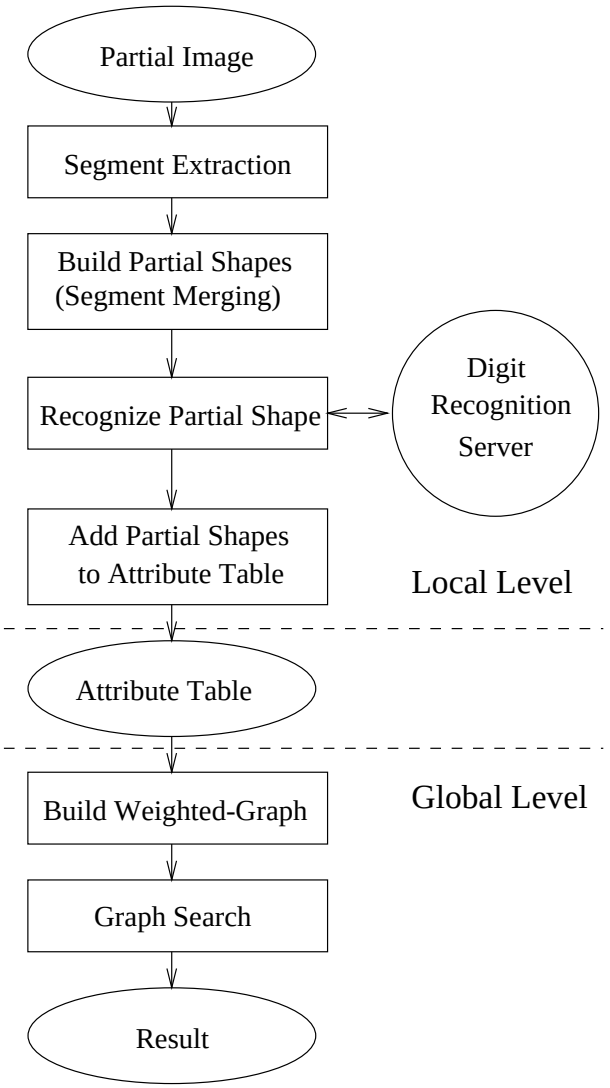


Figure 5: The Segmentation-free Module.

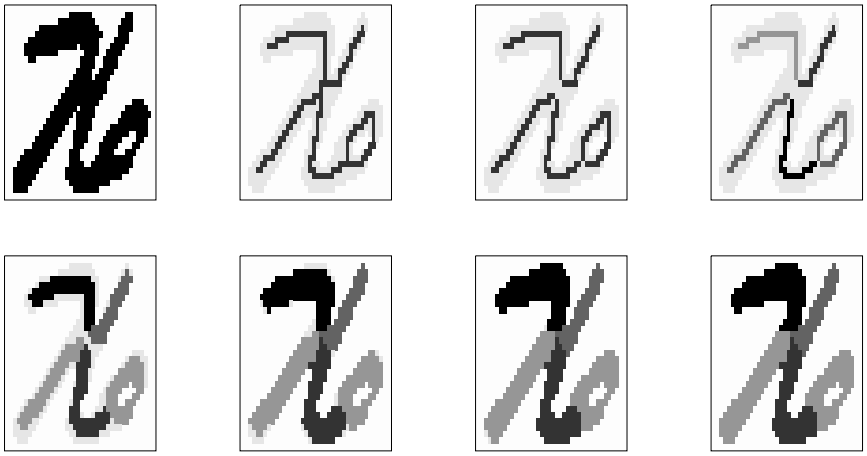


Figure 6: Different steps in segment extraction.

example of accepted PS's.

In the first step the 'Segment Extraction' extracts and orders the *segments* from a PI. Each segment starts and ends with singular points, namely, *end* points, *T-joints*, and *crossing* points. Each of these singular points represents a potential break point between two digits [21]. For the extraction and ordering of the segments the following steps are carried out

1. the PI is thinned
2. singular points are detected and removed, very small CC's are deleted
3. the resulting CC's are labeled
4. the labeled CC's are expanded within the borders of the original PI with the expanded CC's corresponding to the searched segments
5. all segments that are obtained are ordered from left to right

Figure 6 illustrates these steps. The first image shows the original PI, the second image the thinned PI, the third image shows the PI after removal of singular points and small CC's and the fourth image the labeled CC's. The next four images show the expansion of the labeled CC's within the borders of the original PI (the resulting five segments are marked by different gray levels).

The 'Build Partial Shape' step makes use of these segments to build meaningful PS's, susceptible to represent individual numerals. Theoretically, if N_S is the number of segments, then there are $(2^{N_S}-1)$ possible PS's. In order to avoid this exponential complexity, [21] proposed to use spatial constraints to reduce the number of PS's to $\mathcal{O}(N_S^2)$. The resulting building process works as follows. For each segment s_i, $i = 1, .., N_S$, we build $(N_S - i + 1)$ PS's by starting at s_i and including segments to the right of s_i up to s_{N_S} segments, i.e., the set of PS's starting at s_i is given by $\{\bigcup_{j=i}^{k} s_j, i \leq k \leq N_S\}$. There are in total $N_S + (N_S - 1) + .. + 1 = \frac{N_S(N_S+1)}{2}$ PS's. Moreover, a PS must satisfy both the following size and shape constraints to be meaningful.

1. $\frac{PS_{height}}{PI_{height}} > T_{PS1}; T_{PS1} \in [0.2, 0.4]$

2. $\frac{PS_{width}}{PS_{height}} < T_{PS2}; T_{PS2} \in [4, 6]$

The widths and heights of PS's and PI's are those of the corresponding bounding-boxes. Condition 1 prevents very small PS's from being considered meaningful whereas Condition 2 on aspect ratio prevents too wide PS's from being considered meaningful. Such very wide PS's will be discussed later in Section 5.3 on dummy symbols. Due to the two additional size and shape constraints, the actual total number of meaningful PSs is usually smaller than $\frac{N_S(N_S+1)}{2}$.

In the 'Recognize Partial Shape' step each meaningful PS is submitted to the 'Digit Recognition' server. The best class k together with the corresponding score value $q_k(PS)$ and the coordinates of the PS are then added to the 'Attribute-Table' (see Fig. 5). Finally the score S_{PS} for each meaningful PS is computed (see Eq. 4 below) and added to the 'Attribute-Table' for global level processing. The score of a PS S_{PS} represents its quality and is determined by the two factors S_{digit} (see Eq. 1) and S_{emb}.

$$S_{PS}(PS) = S_{digit}(PS) \times S_{emb}(PS) \tag{4}$$

The factor S_{emb} reflects how well the PS is embedded in the PI and is defined by:

$$S_{emb}(PS) = \frac{1}{1 + (\frac{PS_{median} - PI_{median}}{PI_{height}})^2} \tag{5}$$

5.2 Global Interpretation

At the 'Global Level', the system attempts to determine the most consistent combination of PS's. All possible combinations are expressed by means of a directed and weighted graph which is generated in the 'Build Weighted-Graph' step from the 'Attribute-Table'. Therefore the choice of the most consistent combination becomes a graph search problem. In order to build a directed and weighted graph we have to define its nodes and edges with the corresponding edge costs.

The *nodes* of the graph are represented by the meaningful PS's found in 'Local Level' processing. Furthermore there are two special nodes, called Root and End. Two partial shapes PS(i) and PS(j), $1 \leq i < j \leq N_{PS}$, are linked together if they satisfy the following conditions (see Fig.7).

1. $[e(i) = s(j)] or [e(i) + 1 = s(j)]$,
   where $e(i)$ is the index of the end segment of PS(i) and s(j) is the index of the start segment of PS(j). That is, PS(i) and PS(j) share either none or one common segment.

2. PS(i) $\not\subset$ PS(j). This is trivial because otherwise the link does not make sense.

3. $\frac{ox(i,j)}{PS_{height}(i,j)} < T_{overlap1}; T_{overlap1} \in [0.5, 0.7]$,
   where $PS_{height}(i,j)$ is the height of the bounding-box of the union of $PS(i)$ and $PS(j)$, for $ox(i,j)$ see Fig. 7. That is, $PS(i)$ and $PS(j)$ should not too much overlap horizontally.

4. $\frac{oy(i,j)}{PS_{height}(i,j)} > T_{overlap2}; T_{overlap2} \in [0.2, 0.4]$,
   That is, $PS(i)$ and $PS(j)$ should have some vertical overlap.

The above constraints for possible *edges* ensure that all segments of the PI are used in every possible path from Root to End. Root and End nodes are especially treated so that they are connected to the left- and right-hand-side of the numeral string.

The *edge cost* between two linked nodes i and j, $C_{edge}(i,j)$ are determined by the costs of the nodes C_{node} and the costs of the link C_{link} between the two linked nodes The parameter $W_{node} \in [0.4, 0.6]$ adjusts the weight of node costs compared to link costs.

$$C_{edge}(i,j) = W_{node} C_{node}(i,j) + (1 - W_{node}) C_{link}(i,j) \tag{6}$$

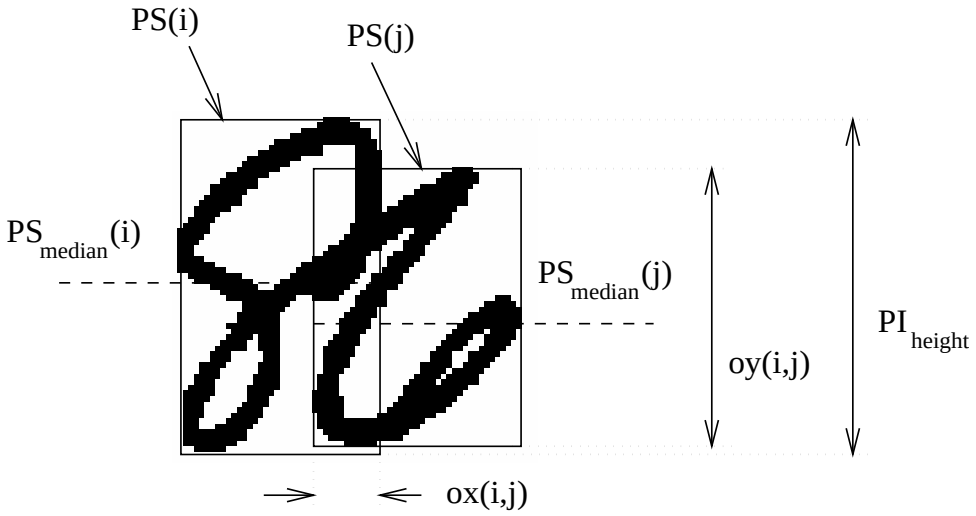


Figure 7: Definition of Relationships between 2 Partial Shapes.

$$C_{node}(i,j) = C_{PS}(i) + C_{PS}(j) \tag{7}$$

$$C_{link}(i,j) = C_{ho}(i,j) + C_{va}(i,j) \tag{8}$$

The costs C_{PS} for the nodes correspond to the sum of the costs for the linked PS's which are defined via their score S_{PS} (see Eq. 4) by the following equation:

$$C_{PS}(i) = \frac{1}{1 + S_{PS}(PS(i))} \tag{9}$$

The edge costs are defined by the sum of the costs for horizontal overlapping C_{ho} of the two linked PS's and the costs for their vertical alignment C_{va} by the following equations (see Fig. 7 for the definitions of various quantities):

$$C_{ho}(i,j) = 1 + 100\frac{ox(i,j)}{min[PS_{width}(i), PS_{width}(j)]} \tag{10}$$

$$C_{va}(i,j) = 1 + 100[\frac{PS_{median}(i) - PS_{median}(j)}{PI_{height}}]^2 \tag{11}$$

All the above expressions are defined such that if PS(i) and PS(j) are each well recognized by the isolated numeral recognizer, have no horizontal overlap, and are vertically well aligned with respect to each other as well as with respect to the partial image, then the edge cost $C_{edge}(i,j)$ is minimum. The *cost of a path* between two nodes is defined as the sum of partial edge costs.

Finally, in the 'Graph Search' step a best-first graph search algorithm is used to find for the path with the lowest cost from the Root to the End node [26]. If no path exists, the PI is

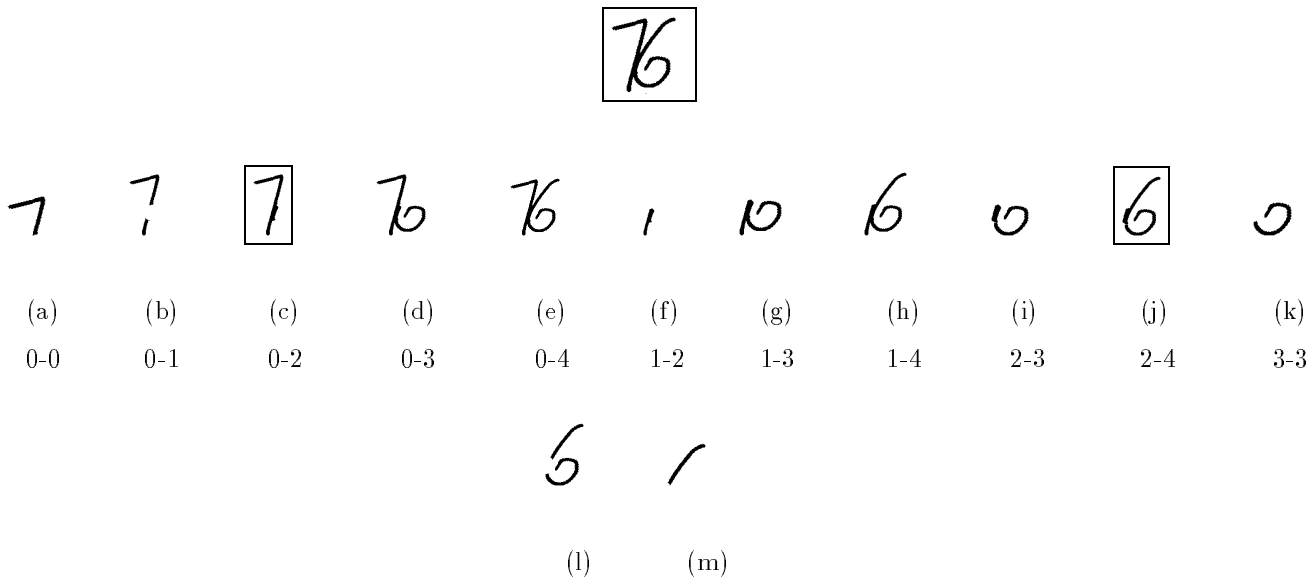
rejected, otherwise an interpretation of the PI is found because each of the nodes on the path represents a recognized digit. For all nodes (without Root and End node) on the path with the lowest costs the 'Digit Recognition' server recognizes the corresponding PS's. Like the 'Digit Detection' module the resulting classes together with the corresponding score values are sent to the 'Global Decision' module.

Fig. 8 illustrates the recognition of a PI. There are 13 meaningful partial shapes: (a), (b), ... and (m), the links between which are shown on the weighted-directed-graph. Note that the two nodes (f) and (g) have no actual predecessors because none of their potential predecessors could satisfy the spatial constraints. Let us consider node (f), which has 6 potential predecessors, namely, Root, (a), (b), (c), (d) and (e). It cannot be linked to Root because it would have left some partial shapes, such as (a), out of use. It cannot be linked to (c), (d) and (e), because more than one segment would be used twice, and it cannot be linked with (a) or (b) because condition 3 for potential links would not allow such a high horizontal overlap. In the last step the graph search algorithm finds the path with the lowest cost from Root to End to be Root $\rightarrow$ (c) $\rightarrow$ (j) $\rightarrow$ End, which corresponds to the numeral string '76'.

Compared with the work of [21], our segmentation-free method provides three extensions. First, we use a statistical isolated numeral recognizer instead of a structural one. One inconvenience of this is that for each partial shape, we must reconstruct a binary image (see Subsection 5.1) before submitting it to the isolated numeral recognizer (see [27] for more detail). The advantage is that our approach is independent of any special digit recognition method and less sensitive to thinning artifacts. Second, we define the score of a PS to be proportional to a weighted sum over all classes instead of using only the best one. Last, many "hard constraints" have been converted into "soft decisions" in the form of costs. The remaining hard constraints are quite loosely set. This allows our system to flexibly accommodate the large spatial variations of totally unconstrained data.

5.3 Dummy Symbol

The goal of the concept of *dummy symbol* is to introduce a systematic way to handle 'noisy' parts of the image that cannot be eliminated by standard filtering algorithms. Examples of such



(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)
0-0	0-1	0-2	0-3	0-4	1-2	1-3	1-4	2-3	2-4	3-3

(l)	(m)
3-4	4-4

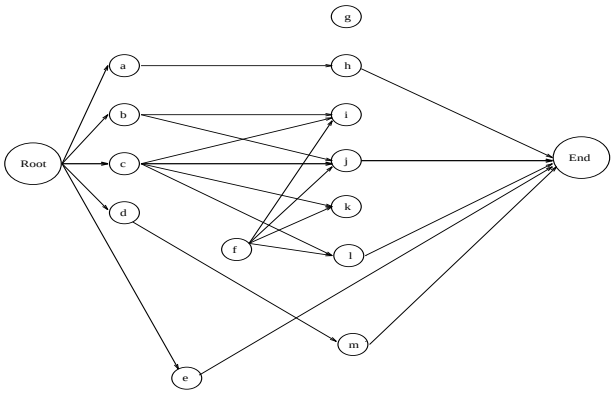


Figure 8: A PI with accepted PS's and the corresponding weighted graph of a numeral string.

noisy parts are hyphen signs, ligatures (such as additional horizontal bar between two zero's, see Fig. 9), underscores and some preprinted graphics as shown in Fig. 1. In this paper, we limit our discussions to the first three types of noisy parts. The use of dummy symbols implies two modifications at two distinct points in the 'Segmentation-free' module.

First, the detection of potential dummy symbols is performed in the 'Building Partial Shape'. A PS is deemed potential dummy if it satisfies both of the following conditions:

1. It is composed of a single segment.
2. $\frac{PS_{width}}{PS_{height}} > T_{dummy}; T_{dummy} \in [4, 6]$

It is clear that these two conditions are necessary for a PS to be a hyphen sign, a ligature or an underscore. They are not sufficient because a PS satisfying them can be part of a valid numeral (e.g., horizontal bar of the number '5' or '2'). Note that the above two conditions are sufficiently strict that no valid numerals can be detected as dummy. Potential dummy PS's are not submitted to the isolated numeral recognizer; they are marked as potential dummy and directly added to the 'Attribute-Table'.

The second modification to deal with dummy symbols is made in the 'Build Weighted-Graph' step (see Fig. 5). Let us consider a PI composed of $N_S = 3$ segments yielding a total of $\frac{N_S(N_S+1)}{2} = 6$ PS's, where the fourth has been detected as a potential dummy symbol (see Fig. 9). The conditions for setting a jump from PS(i) to PS(j) are:

1. e(i) + 2 = s(j), where e(i) is the index of the end segment of PS(i) and s(j) the index of the start segment of PS(j).
2. the PS which consists of the segment with the index e(i) + 1 is marked as potential dummy.

In the example the thick arrow in Fig. 9 indicates the additional edge we have introduced in order to allow a path from node (a) to node (f) by jumping over the potential dummy symbol (d). PS (d) is marked as potential dummy and does therefore not appears in the directed and weighted graph. The cost C_{dummy} of this special edge is given by

$$C_{dummy}(i, \lambda, j) = W_{dummy1} + W_{dummy2} C_{edge}(i, j) \tag{12}$$

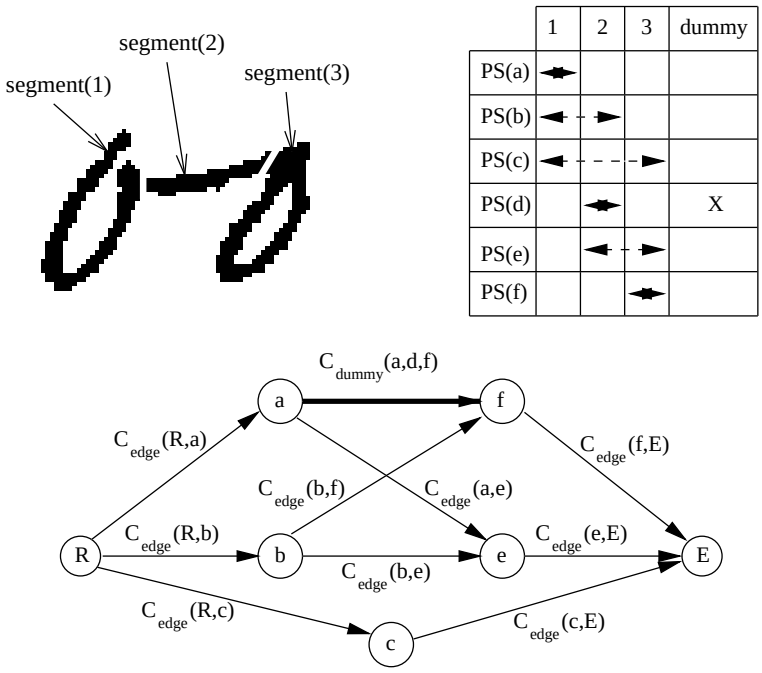


Figure 9: Example of Dummy Symbol Processing.

where i and j are the indexes of the left, respectively right PS of the dummy symbol λ. W_{dummy1} and W_{dummy2} are experimentally determined to be 2 and 20.

In brief, the concept of dummy symbol provides us with the possibility to leave out extraneous parts from the numeral strings. However, this possibility becomes effective only if the removal yields a much lower cost so that the path through the dummy symbol has the lowest cost. Note that the 'Graph Search' procedure remains unchanged.

6 Global Decision

The 'Global Decision' module decides either to accept the recognition result or to reject the SI via a rejection mechanism. The goal of a rejection mechanism is to minimize the number of recognition errors for a given number of rejections. The rejection rule that optimizes the error-reject tradeoff was discovered by Chow in 1957 [4]. Unfortunately the hypotheses underlying the optimal rule do not fit our problem. The most important mismatches are the hypotheses of a finite number of classes and the capability to compute the class posterior probabilities. Therefore

we have developed a heuristic rule based on some preliminary experiments and observations.

For our rejection rule it is necessary to define a score value S_{string} for the whole string. Since a string is considered as misrecognized if any of its digits is misrecognized, it is sensible to assign the lowest score q_k (see Section 4) among all scores to the whole string.

$$S_{string} = \min_i q_k(i) \tag{13}$$

The score values $q_k(i)$ are stored in the result table containing the recognition results from the 'Digit Detection' and the 'Segmentation-free' module (see Table 1 at the end of this section for an example).

Experiments have shown that SI's that do not make use of the 'Segmentation-free' module are relatively easy to segment and can be recognized with a high reliability. Images which use the 'Segmentation-free' module are in general more difficult to segment and therefore recognition of these images is less reliable. This observation suggests that the rejection threshold for the score value S_{string} should be higher when the 'Segmentation-free' module is used. Let $T_{reject1}$ be the rejection threshold when the 'Segmentation-free' module is used and $T_{reject2}$ the rejection threshold when it this is not the case. Reporting both thresholds to a single rejection parameter t according to Fig. 10 results in the following global rejection rule.

IF ('Segmentation-free Module' was used) THEN
  Reject SI, if $S_{string} < T_{reject1}(t)$
ELSE
  Reject SI, if $S_{string} < T_{reject2}(t)$

where $t \in [0, 1]$ and $T_{reject1}(t) \geq T_{reject2}(t)$ reflecting that when the 'Segmentation-free' module is used, the decision is more conservative. In Fig. 10, t is set to A and the resulting threshold values are B and C respectively.

For $t < P$, $T_{reject2}(t) = 0$ and all SI's that do not make use of the 'Segmentation-free' module are accepted. The best value for P is determined experimentally during the training phase of the system and found in the interval $[0.8, 0.9]$. For a given rejection parameter t we can measure the number of rejections and recognition errors using a set of SI's. By varying the rejection parameter

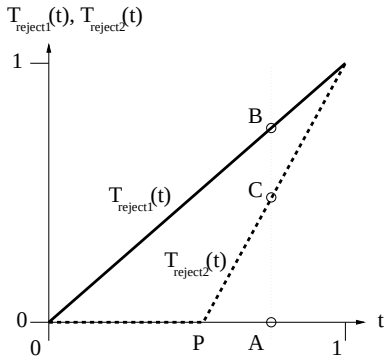


Figure 10: Threshold functions $T_{reject1}(t)$ and $T_{reject2}(t)$

origin	DD	DD	SF		
i	1	2	3	4	5
$k(i)$	'0'	'1'	'6'	'6'	'2'
$q_k(i)$	0.962	0.933	0.591	0.976	0.985

Table 1: Example for a result table from a SI with truth value '01062'.

t over the whole interval $[0, 1]$ error-reject tradeoff curves can be determined experimentally (see Section 7 for some examples).

Table 1 shows an example for a result table. The first line contains the origin of the digit results ('DD' stands for 'Digit Detection' and 'SF' for 'Segmentation-free' module), the second line the digit position i, the third line contains the class $k(i)$ and the fourth line the corresponding score value $q_k(i)$.

By concatenation of the classes $k(i)$ we get the recognition result '01662' with the corresponding score $S_{string} = 0.591$, which is compared to $T_{reject1}(t)$ because the 'Segmentation-free' module is involved. Depending on the rejection parameter t the SI would either be rejected or a recognition error would occur.

7 Experiments and Results

To evaluate the performance of our system, we have carried out experiments with the CEDAR and the NIST SD3 databases. For all reported results we used the definitions for the *recognition rate*, the *error rate* and the *rejection rate* which have been proposed by [5]. Let A be a test set with N_A string images (SI's). If the recognition system rejects N_{rej} images, classifies correctly N_{cor} images and misclassifies the remaining N_{err} i.e. $N_{cor} + N_{err} + N_{rej} = N_A$, then

$$recognition\ rate = 100\frac{N_{cor}}{N_A}\% \tag{14}$$

$$error\ rate = 100\frac{N_{err}}{N_A}\% \tag{15}$$

$$rejection\ rate = 100\frac{N_{rej}}{N_A}\% \tag{16}$$

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

The knowledge about the number of numerals composing a SI was not used in any of our experiments. A SI was counted as correctly classified if all numerals composing it were correctly classified (according the assigned truth value). For each experiment a table summarizes the performance on zero-reject level and the three error levels 2%, 1% and 0.5%. Error-rejection curves demonstrate the performance over the whole rejection range.

7.1 Experiments on the CEDAR Database

The CEDAR database resulted from a project by the *United States Postal Services* (USPS) and *Center of Excellence for Document Analysis and Recognition* (CEDAR) [25]. Images of hand written address blocks from live mail have been gathered at the Buffalo USPS MSC (near New York) in the years between 1986 and 1988 (see Fig. 1 for some examples). The gathered images have been segmented into words, SI's (zip codes), digits and alpha numeric characters. The provided images are divided into a training and a testing set. In our experiments we used SI's from the training set for the optimization of the parameters (see below for further specification). All provided SI's from the test set were used in order to determine the recognition, error and rejection rates.

digits	images	zero	2%	1%	0.5%
5,9	495	83.6	60.0	51.5	48.0
5	436	84.9	64.0	54.5	50.0

Table 2: Recognition rates for the CEDAR experiment for test/binzips

Prior to presegmentation, the skew of the SI was detected and corrected using principal component analysis [28].

The 'Digit Recognition' server has been trained with the 18468 digits from the *train/bindigis* directory. Its performance at zero-rejection level was 98.69% (tested with the 2213 digits provided in the *test/bindigis/goodbs* directory). The parameter for the proposed system have been optimized on the first 500 images in the *train/zipcodes/br* directory.

We used the 495 SI's provided in the *test/binzips/bs* directory as a first test set. Table 2 summarizes the recognition rates for different error levels. The column 'digits' describes the number of digits in the SI, the column 'images' shows the size of the according test sets. The column 'zero' gives the recognition rate at zero-rejection level, the columns '2%', '1%' and '0.5%' reports the recognition rates for the according error levels. In order to achieve an error rate of 1% for SI's containing 5 digits, the system has to reject 44.5% of the images. The remaining 54.5% are correctly classified.

Figure 11 shows the performance for all rejection rates. The curve labeled '5- and 9-digit' shows the performance over all 495 SI's. The curve labeled '5-digit' corresponds to the subset with 436 SI's containing only five digits.

As the second test set we used the 435 SI's provided in the *test/zipcodes* directory. The results of this test are summarized in table 3 and figure 12.

7.2 The NIST SD3 Database

The SD3 database was provided by the American *National Institute of Standards and Technology* (NIST) in 1992 as part of a conference to assess the state-of-the-art in isolated handwritten character recognition [29]. The database contains isolated digits, upper- and lower-case letters

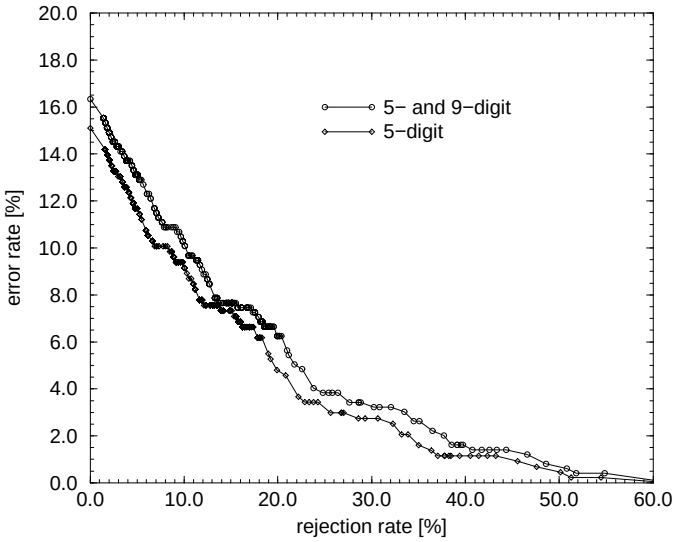


Figure 11: Error rate vs. rejection rate for the CEDAR database, for test/binzips

digits	images	zero	2%	1%	0.5%
5,9	435	72.9	49.5	44.5	43.0
5	398	74.9	51.0	47.5	45.0

Table 3: Recognition rates for the CEDAR experiment for test/zipcodes

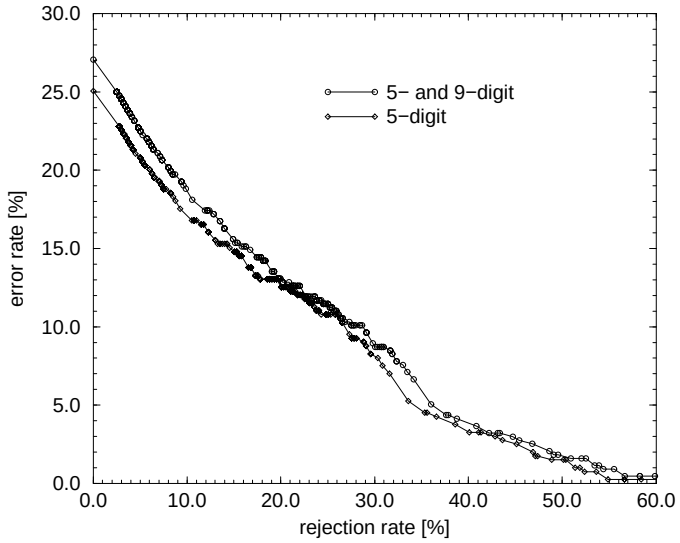


Figure 12: Error rate vs. rejection rate for the CEDAR database, for test/zipcodes

and 2100 scanned images of specially designed forms containing fields for SI's and a short text. The forms have been filled out by employees of the *U.S Bureau of the Census*. The source code to read and registrate the forms and to extract the fields containing the SI's was provided by NIST on the *hsfsys* CD-ROM. On these extracted fields we carried out our experiments on the SD3 database (see Fig. 2 for some examples).

In the experiments on the NIST database, the recognition system has shown better recognition performance without the concept of the dummy symbol. The SI's of the NIST database contain much less noisy parts than the SI's from the CEDAR database. In some cases essential parts of numerals are classified as dummy symbols and a correct interpretation of the corresponding partial image is no longer possible .The reported results correspond to the experiments without the dummy symbol.

The isolated digit recognizer has been trained with the first 50000 digits in the *SD3/data* directory. Its performance at zero-rejection level was 99.45% (tested with the last 173124 digits provided in the *SD3/data* directory). The parameters for the proposed system have been

digits	images	zero	2%	1%	0.5%
2	981	96.2	94.5	93.5	91.5
3	986	92.7	86.0	79.5	70.5
4	988	93.2	86.0	81.0	70.0
5	988	91.1	81.0	77.5	70.5
6	982	90.3	80.5	75.5	66.5
2-6	4925	92.7	86.0	82.0	74.0

Table 4: Recognition rates for the NIST experiment

optimized on 2397 images from the files f1000 to f1099, without the file f1014 [3].

As test set we have selected the files f1800-f1800 and f2000-f2099, which have also been used as a test set by [15]. The file f1823 was not used due to a form registration failure. Another 50 fields were not included in the test. Among these fields were empty fields (3 cases), fields with field extraction errors (36 cases) and fields containing wrong numbers, or numbers which are not recognizable even for humans (11 cases).

Figure 13 plots the error-reject tradeoff for all used fields in the test set. Figure 14 show the behavior for different string length (from two to six digits).

8 Discussion and Comparison

The introduction of the 'Presegmentation' and the 'Digit Detection' modules has proven powerful. These two modules try to recognize all simple cases of numeral strings (not much noise, no touching digits, digits not broken into a lot of CC's, digits recognizable with a high confidence) without using the 'Segmentation-free' module. The experiments on the NIST database have shown that 83% of the images in the test set have been considered as "simple" cases. The recognition rate for this subset was 97.1% compared with 92.7% for the whole test set. On the CEDAR database 68% from the *test/binzips* and 59% of the *test/zipcodes* have been considered as

[3] For file f1014 the assigned truth values did not correspond to the form

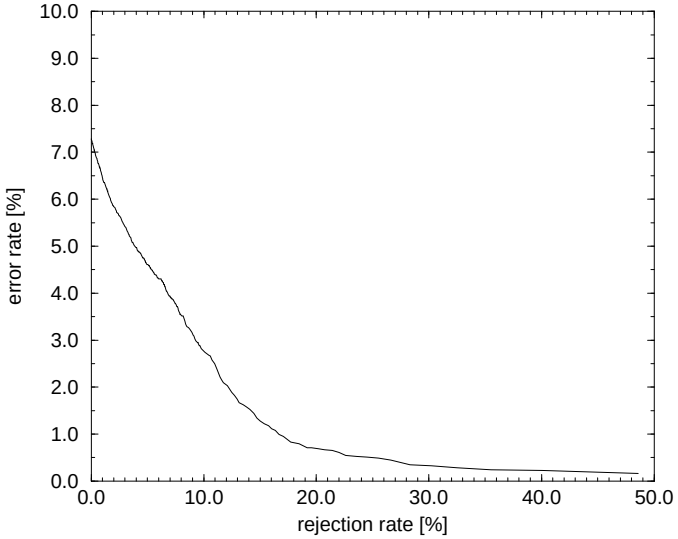


Figure 13: Error rate vs. rejection rate for the NIST database, overall performance

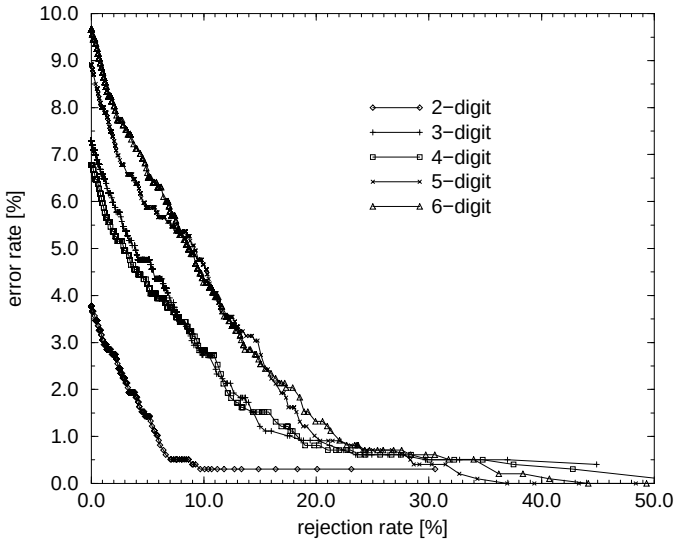


Figure 14: Error rate vs. rejection rate for the NIST database, performance for different length of strings

"simple" cases. The corresponding recognition rates were 92.9% and 89.1%, respectively. There exists a strong relation between the percentage of the "simple" cases and the overall recognition performance.

The proposed system provides a framework to combine segmentation-based and segmentation-free recognition methods. Any method can be used for the 'Digit Recognition' server and any method for the 'Segmentation-free' module.

The reported results on the CEDAR and the NIST SD3 databases have been carried out with an identical recognition system with the following three exceptions: the optimization was carried out on different training sets, the dummy concept is not used for the NIST database and skew-normalization is carried out for the CEDAR database.

Comparison with published methods is delicate in some cases because of the uncertainty concerning the exact data being used. For publications using the CEDAR database we would like to mention the following for which the experimental conditions are comparable. [24] used the data provided in the *test/binzips* directory set but excluded 16 difficult cases in their experiments; they reported a recognition rate of 73.9% for 2.9% error rate. [11] reported for the 495 images in the *test/binzips* a recognition rate of 56% for 0.8% error rate. For the 435 images in the *test/zipcodes* a recognition rate of 42% for 1.2% error rate is reported. The corresponding recognition rates achieved by our system are 51% and 46%, respectively.

For the publications using the NIST database a more detailed comparison is possible. Table 5 summarizes the recognition rates for different error levels for the papers [15], [10] and [14]. [15] used the files f1800 - f1899 and f2000 - f2099 for testing. [10] and [14] did not further specify the used data. A '-' indicates that no recognition rate has been reported for the specified rejection rate or error rate. For [15] and [14] the recognition rate could not be exactly calculated (from published curves) due to a different error rate definition. Therefore the according error rates are provided in parentheses.

In [21], loosely constrained data were used. The authors collected isolated numerals from 260 persons who were asked to write normally and neatly according to shown standard shapes. Numeral strings were then artificially generated by concatenation. They reported correct recognition rates varying from 99% for isolated numerals down to 81.96% for strings composed of 5

authors	digits	images	zero	2.0%	1.0%	0.5%
[15]	2	1000	-	-	87.1 (0.9)	74.6 (0.4)
	3	1000	-	-	84.2 (0.9)	70.7 (0.4)
	4	1000	-	-	76.2 (0.8)	68.7 (0.3)
	5	1000	-	-	70.3 (0.7)	55.7 (0.3)
	6	1000	-	68.6 (1.4)	62.4 (0.6)	57.7 (0.3)
[14]	2	1000	-	96.5 (2.0)	95.2 (1.0)	94.5 (0.5)
	3	1000	-	92.1 (1.9)	88.0 (0.9)	82.6 (0.4)
	4	1000	-	84.3 (1.7)	80.7 (0.8)	75.6 (0.4)
	5	1000	-	81.3 (1.7)	78.6 (0.8)	70.7 (0.4)
	6	1000	-	77.4 (1.6)	70.5 (0.7)	43.8 (0.2)
[10]	5	2000	83.1	69.6 (2.0)	64.3 (1.0)	61.4 (0.5)

Table 5: Recognition rates on the NIST database reported by other methods

numerals, without using the knowledge about the actual length.

No system was applied to both the CEDAR and NIST, or at least no recognition rates have been reported. Compared to the recognition rates provided in Table 5 our system achieves in general better recognition rates for low rejection levels and better recognition rates for strings containing more than three digits.

9 Conclusion

We have proposed a system for off-line recognition of handwritten numeral strings. The system is built upon a number of components, namely, a 'Presegmentation' module, a 'Digit Recognition' server, a 'Digit Detection' module, a 'Segmentation-free' module and a 'Global Decision' module. Experiments on real data from the CEDAR and the NIST SD3 databases showed very good recognition rates at zero-rejection level and a good error/reject tradeoff. In particular, our system which is based on a discrete approach works well on both databases. This is not surprising since

it was originally developed for difficult, totally unconstrained data (CEDAR). One interesting question is to which extent continuous based methods, which are mostly developed for slightly constrained data (NIST), are able to handle totally unconstrained data. Our main results are summarized as follows:

1. The extension of the concept of presegmentation has proven useful. Instead of dividing the input image into individual numerals, we only attempt to divide it into groups of numerals, each consisting of any integer number of numerals.

2. The conversion of "hard constraints" into "soft decisions" allows a more flexible accommodation of large spatial variations in totally unconstrained data.

3. The criterion based on the weighted sum of confidences over all classes has a good discrimination power.

4. The introduction of dummy symbols provides the system with the possibility to not take into account "noisy" parts that cannot be eliminated by standard filtering algorithms.

5. A simple accept/reject rule is suggested which allows the system to make efficient reject decisions for any given error rate or reject rate.

Acknowledgments: This work was partly supported by the Swiss National Science Foundation.

References

[1] F. Arcelli, A. Chianese, M. De Santo, and A. Picariello, "Approaching Character Segmentation with Parallel Contour / Skeleton Analysis", *Proc. of The 8th Scandinavian Conf. on Image Analysis,* Tromsö, Norway, May 25-28, pp. 1299-1306, 1993.

[2] Thien M. Ha, D. Niggeler, and H. Bunke, "A System for Segmenting and Recognising Totally Unconstrained Handwritten Numeral Strings," Proceedings of the third International Conference on Document Analysis and Recognition, Aug. 14-16, 1995, Montréal, Canada, pp. 1003-1009.

[3] R.G. Casey and E. Lecolinet, "Strategies in Character Segmentation: A Survey," Proceedings of the third International Conference on Document Analysis and Recognition, Aug. 14-16, 1995, Montréal, Canada, pp. 1028-1033.

[4] C.K. Chow, "An Optimum Character Recognition System Using Decision Functions," *Institute of Radio Engineers (IRE) Transactions on Electronic Computers* ***6***, pp. 247-254, December 1957.

[5] C.K. Chow, "On Optimum Recognition Error and Reject Tradeoff", *IEEE Transactions on Information Theory,* Vol. IT-16, No. 1, pp. 41-46, January 1970.

[6] H. Fujisawa, Y. Nakano, and K. Kurino, "Segmentation Methods for Character Recognition: From Segmentation to Document Structure Analysis", *Proceedings of the IEEE,* Special Issue on OCR, Vol. 80, No. 7, pp. 1079-1092, July 1992.

[7] J. D. Keeler, D. E. Rumelhart, and W. K. Leow, "Integrated Segmentation and Recognition of Hand-Printed Numerals", in *Advances in Neural Information Processing Systems 3*, R. P. Lippmann, J. E. Moody, D. S. Touretzky (Eds.), Morgan Kaufmann Publishers, pp. 557-563, 1991.

[8] F. Kimura, Y. Miyake, and M. Shridhar, "Handwritten ZIP Code Recognition Using Lexicon Free Word Recognition Algorithm," Proceedings of the third International Conference on Document Analysis and Recognition, Aug. 14-16, 1995, Montréal, Canada, pp. 906-910.

[9] E. Lethelier, M. Leroux, and M. Gilloux, "An Automatic Reading System for Handwritten Numeral Amounts on French Checks," Proceedings of the third International Conference on Document Analysis and Recognition, Aug. 14-16, 1995, Montréal, Canada, pp. 92-97.

[10] A. Filatov, A. Gitis, and I. Kil, "Graph-based Handwritten Digit String Recognition," Proceedings of the third International Conference on Document Analysis and Recognition, Aug. 14-16, 1995, Montréal, Canada, pp. 845-848.

[11] N.W. Strathy and C.Y. Suen, "A New System for Reading Handwritten ZIP Codes," Proceedings of the third International Conference on Document Analysis and Recognition, Aug. 14-16, 1995, Montréal, Canada, pp. 74-77.

[12] G. Congedo, G. Dimauro, S. Impedovo, and G. Pirlo, "Segmentation of Numeric Strings," Proceedings of the third International Conference on Document Analysis and Recognition, Aug. 14-16, 1995, Montréal, Canada, pp. 1038-1041.

[13] G.L. Martin, M. Rashid, and J.A. Pittman, "Integrated Segmentation and Recognition Through Exhaustive Scans or Learned Saccadic Jumps," *International Journal of Pattern Recognition and Artifial Intelligence,* Vol. 7, No. 4, 1993, pp. 831-847.

[14] S.W. Lee and E.J. Lee, "Integrated Segmentation and Recognition of Connected Handwritten Characters with Recurrent Neural Network," Proceedings of the third International Conference on Document Analysis and Recognition, Aug. 14-16, 1995, Montréal, Canada, pp. 413-416.

[15] J. Keeler and D.E. Rumelhart, "A Self-Organizing Integrated Segmentation and Recognition Neural Network," in *Advances in Neural Information Processing Systems,* Vol. 4, J.E. Moody, S.J. Hanson, and R.P. Lippmann (Eds.), San Mateo, CA : Morgan Kaufmann, 1992, pp. 496-503.

[16] O. Matan, J.C. Burges, Y. Le Cun, and J.S. Denker, "Multi-Digit Recognition Using a Space Displacement Neural Network," in *Advances in Neural Information Processing Systems,* Vol. 4, J.E. Moody, S.J. Hanson, and R.P. Lippmann (Eds.), San Mateo, CA : Morgan Kaufmann, 1992, pp. 488-495.

[17] C.Y. Suen, C. Nadal, R. Legault, T.A. Mai, and L. Lam, "Computer Recognition of Unconstrained Handwritten Numerals," in *Proceedings of the IEEE, Special Issue on Optical Character Recognition,* T. Pavlidis and S. Mori (Eds.), 1992, pp. 1162-1180.

[18] Thien M. Ha and H. Bunke, "Handwritten Numeral Recognition by Perturbation Method", *Proc. of the Fourth Int. Workshop on Frontiers of Handwriting Recognition,* Taipei, Taiwan, Dec. 7-9, pp. 97-106, 1994.

[19] J. Sch"urmann, "Pattern Classification", *John Wiley & Sons, Inc,* 1996.

[20] P. L. Sparks, M. V. Nagendraprasad and A. Gupta, "An Algorithm for Segmenting Handwritten Numeral Strings", *Second Int. Conf. on Automation, Robotics, and Computer Vision,* Singapore, Sept. 16-18, pp. CV-1.1.1 to CV-1.1.4, 1992.

[21] H. Nishida and S. Mori, "A Model-Based Split-and-Merge Method for Recognition and Segmentation of Character Strings", in *Advances in Structural and Syntactic Pattern Recognition*, H. Bunke (Ed.), World Scientific, pp. 300-309, 1992.

[22] S. Seshadri and D. Sivakumar, "A Technique for Segmenting Handwritten Digits", *Pre-Proc. of the Int. Worshop on Frontiers in Handwriting Recognition III,* Buffalo, New York, USA, May 25-27, pp. 443-448, 1993.

[23] R. Fenrich, "Segmentation of Automatically Located Handwritten Numeric Strings", in *From Pixels to Features III: Frontiers in Handwriting Recognition*, S. Impedovo and J. C. Simon (eds.), Elsevier Science Publishers B.V., Amsterdam, The Netherlands, pp. 47-59, 1992.

[24] F. Kimura and M. Shridhar, "Segmentation-Recognition Algorithm for Zip Code Field Recognition", *Machine Vision and Applications,* Vol. 5, No. 3, pp. 199-210, Summer 1992.

[25] J.J. Hull, "A Database for Handwritten Text Recognition Research," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. 16, No. 5, pp. 550-554, May 1994.

[26] N.J. Nilsson, *Principles of Artificial Intelligence,* Springer-Verlag, 1982.

[27] D. Niggeler, "Handwritten Numeral String Recognition", Master Thesis, Institut für Informatik und angewandte Mathematik, University of Berne, Switzerland, 1994.

[28] A.K. Jain, *Fundamentals of Digital Image Processing*, Prentice Hall, 1989.

[29] R.A. Wilkinson, J. Geist, S. Janet, P.J. Grother, C.J.C. Burges, R. Creecy, B. Hammond, J.J. Hull, N.W. Larsen, T.P. Vogl, and C.L. Wilson, *The First Census Optical Character Recognition Systems Conference,* The U.S. Bureau of Census and the National Institute of Standards and Technology, Technical Report #NISTIR 4912, Gaithersburg, MD, Aug. 1992.