ALCEU DE SOUZA BRITTO JR.

A TWO-STAGE HMM-BASED METHOD FOR RECOGNIZING HANDWRITTEN NUMERAL STRINGS

Curitiba
2001

ALCEU DE SOUZA BRITTO JR.

A TWO-STAGE HMM-BASED METHOD FOR RECOGNIZING HANDWRITTEN NUMERAL STRINGS

Thesis submitted as a partial requirement for the Degree of Doctor of Science from the Pontifical Catholic University of Paraná, Post-Graduate Program in Applied Informatics.

Area of Concentration: Methods and Techniques of Computation

Supervisor: Prof. Dr. Flávio Bortolozzi
Co-supervisor: Prof. Dr. Robert Sabourin

Curitiba
2001

Britto Jr., Alceu de Souza

A Two-Stage HMM-Based Method for Recognizing Handwritten Numeral Strings, Curitiba, 2001.

142pp.

Tese (Doutorado) – Pontifícia Universidade Católica do Paraná. Departamento de Informática.

1.Análise e Reconhecimento de Documentos. 2. Segmentação. 3. Reconhecimento. 4. Verificação. 5. Cadeias Numéricas Manuscritas. 6. Modelos de Markov Escondidos. I.Pontifícia Universidade Católica do Paraná. Centro de Ciências Exatas e de Tecnologia. Departamento Informática II-t

Dedicated to my wife Priscila and my daughters Melina, Alissa and Briani, for having always been by my side, even in the cold Canadian land.

Acknowledgments

I would like to thank Prof. Robert Sabourin and Prof. Flávio Bortolozzi for supervising my work over these last years. More than supervisors they have been friends.

Thanks to Prof. Ching Y. Suen, who has contributed to this work with helpful suggestions.

I would like to thank Frederic Grandidier, Alessandro Koerich and Alessandro Zimmer for the many helpful discussions we had. Particular thanks are due to Christine Nadal and Nadia Benhamed for helping me to organize the numeral string database.

Most of all, thanks to my family, especially to my father Alceu de Souza Britto, my mother Scheila Maria de Souza Britto, my father-in-law Joel Manoel Pereira and my mother-in-law Vilma Pereira, who have always been there for me.

Finally, I wish to thank the Pontifícia Universidade Católica do Paraná (PUC-PR, Brazil), the Universidade Estadual de Ponta Grossa (UEPG, Brazil), the École de Technologie Supérieure (ETS, Canada), the Centre for Pattern Recognition and Machine Intelligence (CENPARMI, Canada), which have supported this work.

Abstract

This thesis presents a segmentation-free method for recognizing handwritten numeral strings, which is composed of two HMM-based stages. The first stage consists of an implicit segmentation process that takes into account some contextual information to provide multiple segmentation-recognition hypotheses for a given preprocessed string. These hypotheses are verified and re-ranked in a second stage by using an isolated digit classifier. This method enables the use of two sets of features and numeral models: one taking into account both the segmentation and recognition aspects in an implicit segmentation based strategy, and the other considering just the recognition aspects of isolated digits. These two stages have been shown to be complementary, in the sense that the verification stage compensates for the loss in terms of recognition performance brought about by the necessary tradeoff between segmentation and recognition carried out in the first stage. A full evaluation of the proposed method has been carried out on isolated digits - handwritten numeral strings of different lengths - and touching digit pairs extracted, all extracted from the NIST database.

A zero-level rejection was used in the experiments. On 10,000 isolated digits, the method achieved an average recognition rate of 98.02%. The experiments on 12,802 handwritten numeral strings of different lengths showed that the use of a two-stage recognition strategy is a promising idea. The verification stage brought about an average improvement of 9.92% on the string recognition rates. The method achieved global recognition rates of 91.57% and 90.48% strings of known and unknown length respectively. On touching-digit pairs, the method achieved a recognition rate of 89.61%.

Resumo

Este trabalho apresenta um método automático para o reconhecimento de cadeias numéricas manuscritas, o qual é composto de dois estágios baseados em Cadeias Escondidas de Markov. O primeiro estágio tem como estratégia o uso de informação de contexto em uma abordagem de segmentação implícita, cujo objetivo é fornecer múltiplas hipóteses de segmentação-reconhecimento para uma cadeia numérica manuscrita devidamente pré-processada. No segundo estágio, estas hipóteses são verificadas através do uso de um classificador de dígitos isolados. Este esquema permite o uso de diferentes conjuntos de características e modelos de numerais: um considerando aspectos referentes aos processos de segmentação e reconhecimento de maneira simultanea; outro, considerando apenas aspectos relacionados ao reconhecimento de dígitos isolados. Estes dois estágios se completam na medida em que a verificação supre a necessidade de serem compensadas possíveis perdas de capacidade de reconhecimento ocorridas no primeiro estágio, oriundas da necessária negociação entre segmentação e reconhecimento. Uma completa avaliação do método foi elaborada através de experimentos com dígitos isolados, cadeias numéricas manuscritas de diferentes tamanhos, e pares de dígitos que se tocam, todos extraídos da base de dados NIST.

Considerando-se um nível de rejeição igual a zero, o método apresentou uma taxa média de reconhecimento de 98,02% para 10.000 dígitos isolados. Os experimentos com 12.802 cadeias numéricas manuscritas de diferentes tamanhos foram realizados valendo-se ou não do conhecimento *a priori* da quantidade de dígitos constituintes. Esses experimentos mostraram que a abordagem de reconhecimento baseada em dois estágios é uma idéia promissora, dado que o uso de um segundo estágio de verificação permitiu um aumento médio de 9.92% nas taxas de reconhecimento. A taxa média de reconhecimento foi de 91,57% para cadeias numéricas de tamanho *a priori* conhecido e de 90.48% para cadeias numéricas de tamanho desconhecido. A taxa de reconhecimento para pares de dígitos que se tocam foi de 89,61%.

Table of Contents

List of Figures

List of Tables

Abbreviations

BPN	Backpropagation Network
BW	Baum-Welch
CC	Connected Component
CEDAR	Center of Excellence in Document Analysis Recognition
CENPARMI	Centre for Pattern Recognition and Machine Intelligence
CI	Contextual Information
COG	Center of Gravity
DAR	Document Analysis and Recognition
DP	Dynamic Programming
DP	Digit Pair
ERIM	Environmental Research Institute of Michigan
FBFE	Foreground Background Feature Extraction
FFE	Foreground Feature Extration
GSC	Gradient , Structural and Concavity
HMM	Hidden Markov Model
HP	Horizontal Projection
HSF	Handwritten Sample Form
Intra-SSV	Intra-String Size Variation
IPTP	Institute for Posts and Telecommunications Policy
KSC	Kimura, Shridhar and Chen
LBA	Level Building Algorithm
LCA	Linear Confidence Accumulation
LSF	Large Stroke Feature
MDI	Minimum Discrimination Information
MKSC	Modified Kimura, Shridhar and Chen
ML	Maximum Likelihood
MLFNN	Multilayer Feed-Forward Neural Network
MMI	Maximum Mutual Information
NIST	National Institute of Standards and Technology
NString_SD19	Numeral String Special Database 19
pdf	probability density function
PI	Partial Image
PS	Partial Shape
RBF	Radial Basis Function
Sbb	string bounding box
SCB	String Context-Based
SCHMM	Semi-Continuous Hidden Markov Model
SD19	Special Database 19
SDNN	Space Displacement Neural Network
SOFM	Self-Organizing Feature Map
SOISR	Self-Organizing Integrated Segmentation and Recognition
SOM	Self-Organizing Mapping
SR	Segmentation and Recognition
TDP	Touching Digit Pair
UDLRH	Up, Down, Left, Right, Hole
VM	Verification Module
VP	Vertical Projection
VQ	Vector Quantization
Vrun	Vertical Run

1. Introduction

In spite of the major effort that has been expended to bring about a paper-free society, a very large number of paper-based documents are processed daily by computers all over the world in order to handle, retrieve and store information. The problem is that the manual process used to enter the data from these documents into computers demands a great deal of time and money.

The field of Document Analysis and Recognition (DAR) has played a very important role in the attempt to overcome this problem. The general objective of DAR research is to fully automate the process of entering and understanding printed or handwritten data into the computer. Many methods for doing this have been explored over the past years by a large number of researchers with a view to making computers process documents reliably. The big challenge is to make computers approach human performance in terms of understanding these documents.

1.1. Problem definition

The focus of this work is the recognition of handwritten numeral strings, an important subject of research in the DAR domain. The principal motivation is the wide variety of potential applications, such as: ZIP codes, bank checks, tax forms and census forms. The challenge is to recognize numeral strings of unknown length which are not neatly written. Some possible difficulties contributing to the unsatisfactory performance of many methods for recognizing handwritten numeral strings are:

Size variation: this can occur among different strings (*inter-string size variation*) or among digits in the same string (*intra-string size variation*). The latter represents a difficult problem for recognition methods in which there is no prior segmentation of strings into digits, since the string is processed as a whole entity. In this case, a size normalization method should be able to deal with both inter- and intra-string size variations at the same time.

Slant variation: this may be considered as a writer's characteristic. However, in writer-independent methods, the slant is considered useless and merely a factor contributing to the script variability. Moreover, in numeral strings, the slant contributes to increase the overlap between adjacent digits.

Broken numerals: this problem is frequently encountered in two-stroke numerals, such as 4 or 5; or it may be caused by poor writing conditions or poor quality of the scanning and binarization processes. Broken numerals represent a difficult problem for segmentation-based recognition methods, in which parts of broken numerals need to be grouped before the recognition process.

Overlapping numerals: adjacent numerals may overlap one other. This problem also contributes to increase the difficulty of segmenting the string into digits, since a segmented subimage may contain part of another numeral.

Touching numerals: this is the most complicated problem for many reasons, among them the existence of different kinds of touching samples. The most common touching situations are: a) Single Point Touching, where two numerals have only one touching point between them; b) Ligature Touching: the touching is caused by extra ligature between adjacent numerals; and c) Multiple Point Touching: where the number of touching points is greater than 1.

Noise: this is an useless pattern in the numeral string which is most often caused by extra information related to poor writing conditions or different handwriting styles.

Figure 1-1 shows some examples of string difficulties.

Size variation	Slanted numerals	Broken numerals	Overlapping numerals	Touching numerals
8201 80 43	0123456789 4234 701	53 430	67 55 40	75 25 488

Figure 1-1 String difficulties

Given these string difficulties, the basic idea of segmenting a string into separate entities representing individual digits prior to a recognition step has frequently become unreliable. In fact, a correct segmentation often depends on a correct recognition. By contrast, a correct recognition of a numeral string of unknown length also requires a correct segmentation. We can say that these statements have a “chicken and egg” relationship. Thus, they should be approached simultaneously.

In this direction, various methods have been based on joining segmentation and recognition processes. Some of these are described in Chapter 2. A promising approach has been the use of an implicit segmentation-based method using Hidden Markov Models (HMMs). This approach was originally developed in the field of speech recognition [RABINER, 1989], where it has been applied with much success. More recently, [BOSE & KUO, 1994] and [ELMS, et al., 1998] have shown the benefits of applying it to recognize printed words. From these works, we may conclude that such an approach is a promising way of integrating segmentation and recognition to deal with the difficulties encountered in processing handwritten numeral strings. However, there is some cost attached to this integration, which is an open problem in such a method. This cost is a loss in terms of recognition performance caused by joining segmentation with recognition. In other words, the problem is that a set of features and models that shows promising performance in terms of segmentation usually does not show similar performance in terms of recognition, and *vice versa*. On the other hand, to integrate them it is necessary to define features and numeral models to contemplate both the segmentation and recognition aspects simultaneously. Moreover, the feature set must be extracted from a numeral string image in the same way that it is from an isolated digit image. In summary, the challenge is to find some way to compensate for the loss in recognition performance resulting from the necessary tradeoff between segmentation and recognition carried out in an implicit segmentation-based method.

The proposed method for recognizing handwritten numeral strings provides a way of joining segmentation and recognition taking into account this necessary tradeoff. The method is based on a two-stage recognition strategy that enables the use of two sets of features and numeral models: one taking into account both the segmentation and recognition aspects in an implicit segmentation-based process, and another considering just the recognition aspects in a further verification process.

The work described in this thesis has been also reported in a number of papers. The proposed slant normalization method for handwritten numeral strings is described in [BRITTO et al., 1999]. Some preliminary analyses using the first stage of the proposed method are presented in [BRITTO et al., 2000]. The contribution to string recognition performance by using an enhanced HMM topology in the Level Building Algorithm (LBA) framework is shown in [BRITTO et al., 2001a]. A general overview of the proposed method and some experiments considering known-length numeral strings are presented in [BRITTO et al., 2001b].

1.2. Research goals

The primary goal of this research is to propose a segmentation-free-based method for recognizing handwritten numeral strings in which a prior segmentation of a string into digits is avoided by the use of an implicit segmentation approach and a further verification step. The idea is to provide a way of obtaining a better compromise between segmentation and recognition. The method we have developed to deal with this challenging problem is based on a two-stage recognition strategy enabling the use of two sets of features and numeral models: one taking into account both the segmentation and recognition aspects, and another considering just the recognition aspects.

The general overview of the proposed method is shown in Figure 1-2. The String Context-Based Stage (SCB) is responsible for finding the *N* best segmentation-recognition paths for a given numeral string. For this purpose, dynamic programming is used to match numeral HMMs against the unsegmented and preprocessed string. The 10 numeral HMMs (l_c^0, l_c^1,..., l_c^9) used in this stage are trained on isolated digits, but they take into consideration contextual information regarding string slant and size variations. In addition, features extracted from the foreground pixels of the string image columns are used for contemplating both the segmentation and recognition processes.

The objective of the Verification Stage is to re-rank the *N* best segmentation-recognition paths provided by the first stage using a powerful isolated digit recognizer. This stage consists of an HMM-based digit classifier trained on isolated digits without taking into account string contextual information. A new set of features combines foreground and background information in order to improve the recognition performance of the numeral HMMs. Moreover, 10 additional numeral HMMs

(l_{vr}^0, l_{vr}^1,..., l_{vr}^9) based on the rows of numeral images are combined with the column-based models (l_{vc}^0, l_{vc}^1,..., l_{vc}^9) during the verification process. This combination of column and row models ensures an accurate representation of the numeral classes.

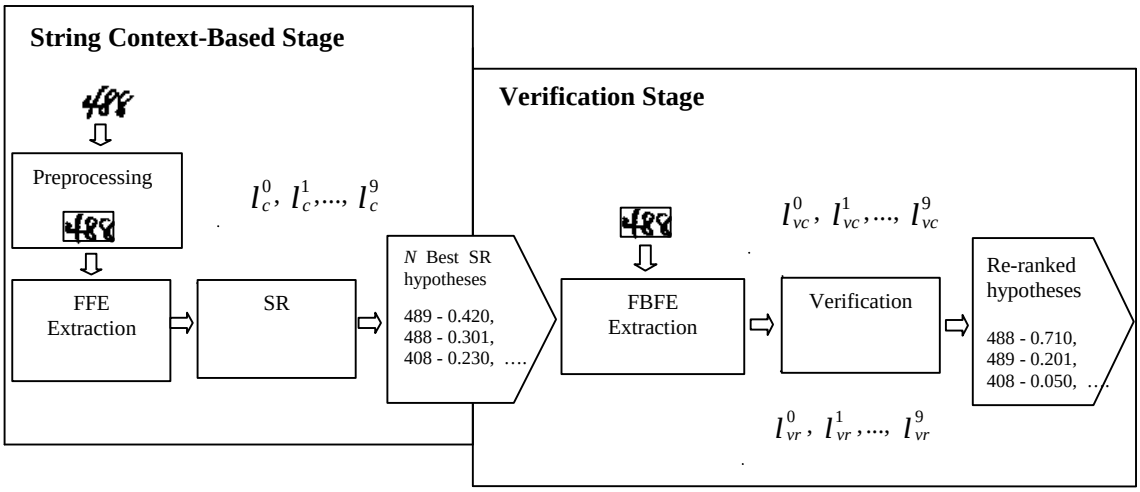


Figure 1-2 General overview of the proposed method

To accomplish the primary goal above described, efforts are geared towards the following:

1. To implement a preprocessing method in order to reduce problems caused by slant and size normalization.
2. To design and implement the String Context-Based (SCB) stage.
   2.1. To design and implement the Feature Extraction Method considering an implicit segmentation-based process.
   2.2. To implement a vector quantization method to produce discrete observations from the feature space.
   2.3. To implement the algorithms for training and testing the HMM numeral models.
   2.4. To design the HMM topology of the numeral models.
   2.5. To train the HMM numeral models.
3. To evaluate the preprocessing method using the SCB stage.
4. To evaluate the contribution to recognition performance by using string slant and size-based contextual information during training of the numeral models.
5. To design and implement the Verification stage.

5.1. To design and implement the Feature Extraction Method considering the recognition of isolated digits.
5.2. To design the HMM topology of the column and row based numeral models.
5.3. To train the column and row HMM-based numeral models.

6. To incorporate SCB and Verification stages into a segmentation-free based method for recognizing handwritten numeral strings.
7. To test the performance on isolated and handwritten numeral strings considering a rigorous evaluation protocol.

A literature review in Chapter 2, and some important observations obtained by analyzing a representative data set extracted from National Institute of Standards and Technology Special Database 19 (NIST SD19) in Chapter 4 support this proposal.

1.3. Contributions

The original contributions of this work can be summarized as follows:

- A two-stage HMM-based method for recognizing numeral strings of unknown length. With this method, a prior segmentation of the string into digits is avoided by the use of an implicit segmentation-based strategy and a further verification step. It allows the combination of different feature sets and numeral models. The verification stage is used to compensate for the loss in terms of recognition performance brought about by integrating segmentation into the recognition process.

- A string slant normalization method, in which we assume that each connected component (CC) in the string has its own slant. The slant and contour length of each CC are used for obtaining the mean slant of the string. The proposed method has shown a positive impact on the number of overlapping numerals in strings.

- The use of slant contextual information, during training of the numeral models, to deal with the significant difference between the slant estimated from isolated digits and that estimated from their original strings. The idea consists of using this contextual information to provide the same conditions during training and testing of

an implicit segmentation-based method. To this end, the slant estimated from the original string is applied to correct the slant of the isolated digits used to train the numeral models.

- The use of contextual information regarding intra-string size variation to deal with the blank space above and below some digits in the string bounding box. This problem cannot be solved by linear size normalization, in which the entire string is reduced to a constant height. Moreover, the use of non-linear size normalization may cause distortions on the digit strokes and broken digits.

- Enhancement of the HMM topology in the LBA framework by adding an end-state to the HMM structure. This additional state brings about a better distribution of the observations among the HMM states. Consequently, the LBA provides a better definition of string segmentation cuts.

- A space model built into the numeral models in order to better represent numeral strings. By constructing the space model inside the numeral models, the problems associated with predicting the number of digits in an unknown-length string in the framework of the LBA are avoided. We have evaluated the use of one space model for each numeral class; and the use of one space model representing all numeral classes. A two-step training mechanism is used in both experiments. This means that the numeral models are first trained on isolated digits. Then, the space model parameters are estimated during the second training step, which is based on digit pairs. The parameters corresponding to the numeral models are kept the same as estimated during the first training step based on isolated numerals.

- The use of a classifier based on the Bayes Theory to predict the string length from the width (number of columns) of the string bounding box. The top 3 hypotheses of the string length predictor are used to reduce the search space in the LBA framework.

- A complete evaluation of the combination of an implicit segmentation method and the use of a verifier to check their segmentation-recognition hypotheses has been conducted by considering isolated digits and handwritten numeral strings of different lengths, as well as a set of touching-digit pairs. The experiments on numeral strings were performed following two strategies: a) an informed strategy, *i.e*. the string length (number of digits) is known. The objective of which was to evaluate the system in different conditions, while at the same time adjusting some important aspects regarding string normalization, feature extraction and HMM parameters; b) a non-informed strategy, where the string length is unknown.

1.4. Outline of the thesis

This document consists of seven chapters and one appendix. In Chapter 2, a review of handwritten isolated numeral recognition and handwritten numeral string recognition is presented, in which the principal investigations that have allowed improvements to be made in the recognition of isolated handwritten numerals are described. Some recent works and the results are presented. Moreover, different approaches to the recognition of handwritten numeral strings are described and examples given. A discussion of these approaches is included.

Chapter 3 introduces the background theory relevant to the handwritten numeral string recognition method described in this thesis. The first topic is related to Hidden Markov Model, since this modeling method is used in both the SCB and verification stages of the proposed method. We present the main HMM concepts, types of HMMs and the algorithms to implement this statistical technique for modeling real problems. These algorithms are related to the process of training and scoring HMMs. In addition, the K-means vector quantization algorithm is described. This algorithm is necessary for mapping our feature space onto the discrete observations used in the proposed numeral HMMs. Finally, a brief review of the Bayes theory is provided. This theory is used to implement a classifier in order to predict the string length (number of digits) from the bounding box width (in columns).

In Chapter 4, the preprocessing techniques used to transfer the handwritten numeral string into a more appropriate form are described. The general idea is to reduce script variability, while specific goals are dependent on the proposed recognition

method. The first topic is related to slant normalization in an implicit segmentation-based method. Different word slant normalization methods are presented and discussed. A word slant normalization method is selected and modified in order to improve the results for handwritten numeral strings. We assume that each connected component (CC) in the string has its own slant. The slant and contour length of each CC are used for obtaining the mean slant of the string. Both the original and modified methods are evaluated by means of some interesting analyses on the NIST SD19 database. These analyses show: a) the positive impact of slant correction on the number of overlapping numerals in strings, and b) the difference in normalizing isolated numerals based on the slant estimated from their own images and the slant estimated from their original string images. A second topic describes a size normalization method which also takes into account an implicit segmentation-based strategy for recognizing numeral strings. This method is based on non-linear normalization and is designed to deal with inter- and intra-string size variations at the same time. However, the recognition results in Chapter 6 have shown that training the numeral HMMs taking into account the intra-string size variation can be used to avoid possible distortions on the digit strokes caused by a non-linear size normalization method. Finally, the method proposed in [SUEN et al., 1992] for smoothing binary images is described. We use it to smooth the string contour in order to reduce the presence of spikes and notches caused by scanning noise or by the slant normalization method. All analyses performed in this chapter support the idea of using contextual information regarding string slant and digit size variations within the string to train the numeral HMMs in the SCB recognition stage.

In Chapter 5, the proposed method, which can be categorized as a segmentation-free approach, is described in detail. This description starts with a general overview, then each stage and the corresponding modules are described. The three modules of the SCB stage are: Preprocessing, Foreground Feature Extraction (FFE) and Segmentation-Recognition (SR). We describe the foreground features and the HMM models used to represent the digit classes in the SCB stage. An end-state in the HMM topology has proved useful for providing a better distribution of the observations among their states. This has brought about an improvement in terms of segmentation performance in the SCB stage. This additional state in the HMM topology also makes it possible to incorporate contextual knowledge regarding strings into the numeral models in the SCB

stage by means of a string-based training method used to build a space model inside of the numeral models. Finally, we present the Verification stage, the purpose of which is to re-rank the best hypotheses generated by the SCB stage. A new set of features and numeral models is defined to ensure an accurate recognition of isolated digits.

Chapter 6 presents a rigorous experimental protocol for implementing and evaluating the proposed string recognition method. Experiments are performed considering isolated digits and numeral strings of different lengths extracted from the NIST SD19 database. In the first experiments, string recognition is based on an informed strategy, *i.e.* the string length (number of digits) is known. The objective of using this strategy is to evaluate the system in different conditions, while at the same time adjusting some important aspects regarding string normalization, feature extraction and HMM parameters. A non-informed strategy is used in the last experiments.

The protocol used to implement and evaluate the proposed method consists of three steps. In the first step, called *SCB Stage Construction and Evaluation*, a baseline system composed of Preprocessing, FFE and Segmentation-Recognition (SR) modules representing the first stage of the numeral string recognition method is evaluated. The Preprocessing module is constructed taking into account the preliminary analyses described in Chapter 4. The Preprocessing and the SR modules are experimentally defined and evaluated. In a second step, called *Verification Stage Construction and Evaluation*, the SCB stage is modified in order to provide the best segmentation-recognition paths or hypotheses. The verification stage is used to re-rank these hypotheses. For this purpose, an isolated digit classifier is developed to check each string segment provided in the segmentation-recognition hypotheses in the SCB stage. In the last step of the evaluation protocol, the system is evaluated using a non-informed strategy, where the string length is unknown. An error analysis is presented and all the experimental results are discussed.

In Chapter 7, the conclusions and some directions for future works are presented, while in Appendix A the processes used for extracting numeral strings from the full-page forms available in the Special Database 19 (SD19) of the National Institute of Standard Technology (NIST) are described. With the handwritten numeral strings extracted from these forms, we create a database called NString_SD19 for training, validating and testing recognizers which go beyond the isolated digit classification.

Moreover, we describe the process used to provide an isolated digit database in which each digit sample has a link with its original string.

2. State of the Art

In this chapter, we address the main topics related to the proposed method: handwritten isolated numeral recognition, and handwritten numeral string recognition. First, we present the main investigations that have made possible improvements in recognition performance of isolated handwritten numerals. Some recent work is briefly described in terms of features, types of classifiers, test databases and results. Moreover, different approaches for recognizing numeral strings are presented and discussed. A general discussion is also presented.

2.1. Handwritten numeral recognition

Many approaches to solving the handwritten numeral recognition problem have been proposed in recent years due to its numerous possible applications. Drawing up a taxonomy of these approaches is difficult, since their methodologies overlap. However, research in this field has basically considered investigating: a) feature extraction methods; b) classification methods; and c) system architectures based on different strategies, such as combinations of multiple classifiers, the use of multiple templates, and the use of verification modules.

The investigation of feature extraction methods has gained considerable attention since a discriminative feature set is considered the most important factor in achieving high recognition performance. In [TRIER et al., 1996] a survey of feature extraction methods for off-line recognition of segmented characters is presented. The authors describe important aspects that must be considered before selecting a specific feature extraction method. Another interesting work of shape analysis techniques can be found in [LONCARIC, 1998].

In general, the feature extraction methods for numeral recognition reported in the literature have been based on two type of features, statistical and structural. The statistical features are derived from statistical distributions of points, such as zoning, moments, projection histograms or direction histograms [KIMURA & SHRIDHAR, 1992][GADER & KHABOU, 1996][CHEUNG & YEUNG, 1998]. Structural features

are based on topological and geometrical properties of the character, like strokes and their directions, end-points, or intersections of segments and loops [PAVLIDIS, 1986][HIRANO et al., 1997][LEE & GOMES, 1997][CAI & LIU, 1998].

Many researchers have explored the integration of structural and statistical information to highlight different character properties, since these types of features are considered to be complementary. In [CAI & LIU, 1998] structural and statistical information is integrated into an HMM-based classifier. The authors use state-duration adapted transition probability distribution and macro-states to overcome the weakness of the HMMs in modeling structural features. Both statistical and structural features are extracted from chain code (locations, orientations and curvatures). The recognition rate is 96.16% in 2,711 digit samples extracted from the CEDAR database.

Another multifeature-based system is proposed in [HEUTE et al., 1998]. In this work, a combination of seven different families of features is proposed in order to arrive at a complete character description. These features are divided into global features (invariant moments, projections and profiles) and local features (intersections with straight lines, holes and concave arcs, extremities, end-points and junctions). A set of 53,324 digits extracted from the NIST database is used to test the system. The recognition, rejection and substitution rates are 90.82%, 8.93% and 0.25% respectively.

Alongside these investigations of feature extraction methods, many other studies have addressed classification methods. Different classifiers have been used for handwritten numeral recognition, such as statistical [GILLOUX, 1994][CHEUNG & YEUNG, 1998][PARK & LEE, 1998], structural [SHRIDHAR & BADRELDIN, 1986][HIRANO et al., 1997] and neural nets [CAO et al., 1995][LIM & CHIEN, 1998][ZHANG et al., 1998]. In recent years, significant contributions to increasing recognition rates have been achieved with different combinations of classifiers [XU et al., 1992][LAM & SUEN, 1995].

An interesting investigation can be found in [KIM et al., 1997]. The authors combine all three possible cases of five kinds of neural network classifiers with different feature sets: gradient, structural, UDLRH (Up Down Left Right Hole), Mesh, and LSF (Large Stroke Feature). Three combination methods are used: majority voting, borda count and LCA (Linear Confidence Accumulation). In the majority voting method, the output of the neural network classifier (real values ranging from 0 to 1) is

transformed into 1 or 0, depending on whether it is the highest output or not. In the borda count method, the output of the neural network classifier is transformed into a decreasing rank order, using:

$$t = 1 - \frac{rank - 1}{\max rank}. \tag{2.1}$$

Finally, in the LCA method, the output of the neural network classifier is used without any transformation. The objective is to determine which subset of classifiers achieves the optimal combination results. The test data for digit recognition are taken from the NIST database. The number of samples used is 10,909. The gradient, structural and UDLRH combination supplies the best results, using a borda count combination method. The recognition, rejection and substitution rates are 98.62%, 0.42%, and 1.39% respectively.

In [HIRANO et al., 1997], a statistical classifier based on local features (distribution of local directions extracted from contours) and a structural classifier based on contour segments and loops are combined. The objective is to compensate for the difficulty of using statistical features in the recognition of patterns, as they are either seriously distorted or similar to one other in shape, or both. First, the statistical classifier recognizes the input pattern. Subsequently, the structural classifier verifies the input pattern as one of two higher ranking candidates acquired by the statistical classifier. The IPTP digit database is used for testing (17,916 samples). The recognition rate is 98.87%.

In recent work, different methods for combining the decisions of two classifiers based on RBF (Radial Basis Function) networks are examined by [CHIM et al., 1998]. The feature set is composed of diagonal and partitioned radial projections, and four-directional edge maps of the image. Four methods for combining the two classifiers are investigated: 1) one classifier is used to process only rejected samples from the other classifier; 2) both classifiers classify each sample independently, and the system assigns a label only when both classifiers agree on its identity; otherwise, the sample is rejected without further verification; 3) the label is assigned as according to the classifier that has the higher output activation level; and 4) both classifiers output each sample independently, and the system assigns the label based on an average Bayes classifier formulation. The highest recognition rate is achieved when one classifier is used to

process only rejected samples from the other (first method). During the experiments, 620 samples of segmented, hand-printed characters written by 8 different people were used. The recognition, rejection and substitution rates are 96.61%, 2.2%, and 1.13% respectively.

In addition, investigations of new strategies for handwritten numeral recognition have led researchers to figure out alternative ways to treat the variability intrinsic to handwritten numerals. In this context, three examples are interesting: 1) In [CHEUNG & YEUNG, 1998] a set of 23 digit prototypes and a deformable model-based recognition in a Bayesian framework are proposed. Without any discriminatory training, they achieved an accuracy of 94.7% with no rejections on a subset of 11,791 images by 100 writers extracted from the NIST database; 2) In [ZHOU et al., 1997] the authors proposed a rule-based structural classifier as a verification module (VM) for a Multilayer Feedforward Locally Connected Network. In their system, two levels of verification are considered: low level (result confirmation by structural features) and high level (cross-check by user-specified rules). Another important aspect is that the VM never changes the result, although it may provide some possibility for rejection repair. To test the system, 18,000 numerals randomly selected from 20 forms written by different users were used. The recognition, rejection and substitution rates are 98.95%, 0.1% and 0.05% respectively; 3) Correia and Carvalho [CORREIA & CARVALHO, 2000] propose an interesting approach for recognition of unconstrained handwritten numerals in which the biorthogonal spline wavelets Cohen-Daubechies-Feauveau (CDF) 3/7 are used as a feature extractor. A multilayer cluster neural network is trained with the backpropagation momentum algorithm. The system is evaluated on 6,000 isolated numerals from the CENPARMI database – 4,000 for training and 2,000 for testing. The recognition, rejection and substitution rates are 94.7%, 1.8% and 3.5% respectively.

Other important contributions in this field are found in Table 2-1. It is important to point out that the recognition, rejection and substitution rates reported are not directly comparable, since these results are based on different databases.

Table 2.1 Other contributions to handwritten numeral recognition

Reference	*Feature extraction method*	*Classification method*	*Test database*	*Recog. rate (%)*	*Rejection rate (%)*	*Subst.. rate (%)*
[FATAVA et al., 1994]	Multiple features (512-bit feature vector) 1) Gradient Features 2) Structural features (micro-strokes) 3) Concavity features	GSC (Gradient, Structural, Concavity) classifier	2,700 digits from various databases	98.87	-	2.13
[GADER, 1996]	1) Automatic feature generation guided by two different evaluation measures: orthogonality and information. 2) Down-sampled normalized image (12x9) using local averaging in 2x2 regions. 3) Transition features (background pixels to foreground pixels) 4) local, two-dimensional convolutions	Four neural networks, one for each set of features. The outputs of these classifiers are averaged.	10,000 digits from ERIM (Environmental Research Institute of Michigam)	98.0	-	2.0
[LIN et al., 1997]	1) 288 statistical features reduced to 32 features using principle component analysis 2) 20x20 binary image	Two neural networks: 1) Backpropagation network (BPN) 2) Self-organizing mapping (SOM) A third classifier (BPN) is used to combine the results	3,000 digit samples from NIST database	98.24	0.0	1.76
[TEO & SHINGHAL, 1997]	1) Structural features extracted by decomposing the binary image into nodes (loops, paths, threads) 2) 16x16 binary image	Hybrid classifier 1) Rule based classifier 2) 8 neural nets, one for each candidate set, which are composed of 3 classes. The rule based classifier outputs potential digit classes. A potential class Ci is chosen. The neural nets that correspond to the candidate sets where Ci appears are invoked. The recognition result is defined using the highest confidence value output by the neural nets.	20,000 digits from NIST database	93.21	3.47	3.26

Table 2-1 – Other contributions to handwritten numeral recognition (Cont.)

Reference	*Feature extraction method*	*Classification method*	*Test database*	*Recog. rate (%)*	*Rejection rate (%)*	*Subst. rate (%)*
[LEE & GOMES, 1997]	1) Structural features (cavities, crossing sequences, number of intersections with principal and secondary axis) 2) Statistical features (pixel distribution) 3) Binary image after scale normalization, thinning and elimination of spurious segments	1) Topological feature classification (rule based classifier) 2) Neural classification (Four Hopfield neural networks used in two steps)	606 numeral images extracted from 121 Brazilian bank checks	92.4	-	-
[KIMURA et al., 1998]	Gray scale pattern obtained by counting the number of black pixels in 10x10 blocks of binary numeral image.	Auto-associative Neural Networks (one for each digit class)	14,979 samples collected by IPTP	98.11	-	-
[ZHANG et al., 1998]	Binary image	Elastic Net Models (one for each digit class). A classifier is obtained by using a decision module which compares the distance between the reconstructed vector and a model. A minimum operator is used to associate the class of a model with the smallest error.	10,000 digits from NIST database	95.7	-	-
[CAI & LIU, 1998]	Statistical and structural features extracted from the chain code. (locations, orientations and curvatures)	HMMs (one model per class)	2,711 digits from CEDAR CDROM1	96.16	-	-
[DELEVSKI & STANKOVIC, 1998]	Morphological and Topological properties Digits are topologically represented by graphs, and morphological properties of those graphs are extracted (forks, joints, relative branch lengths, branch angles, branch positions, *etc.*)	Search for a graph which better represents the incoming digit.	2,213 samples from CEDAR database	99.37	0.27	0.36

2.2. Handwritten numeral string recognition

Another important subject of research in the document analysis and recognition field has been the recognition of numeral strings. Different approaches have been proposed in the literature to deal with this challenging problem. Usually, the taxonomy for these methods takes into account the strategy employed to recognize the string, top-down or bottom-up.

In methods using a top-down strategy, also called holistic methods, string recognition is performed considering the whole string, without *a priori* segmentation into digits or small fragments. The price of avoiding the segmentation problem is to constrain the string recognition system to a limited lexicon. Usually this kind of approach has been used for the handwritten recognition of cursive words (for example, city names or street names in mail address recognition). An example of a holistic method for the recognition of touching-digit pairs can be found in [WANG et al., 1998].

On the other hand, in methods using a bottom-up strategy, also called analytic methods, a numeral string is recognized from its components, such as digits or primitive segments. These methods require either an explicit or an implicit segmentation process. A survey of segmentation strategies is provided by [CASEY & LECOLINET, 1996], where the analytic methods are divided considering the strategy used for string segmentation:

- The dissection strategy, which consists of segmenting the numeral string image into meaningful components (individual numerals) based on "numeral-like" properties [SHI et al., 1997][NISHIWAKI & YAMADA, 1998]. Generally this is a difficult task, due to the possible numeral string difficulties described in Chapter 1.
- The recognition-based strategy, which searches the numeral string for components that match numeral classes. In other words, isolated numeral models are matched against an unsegmented numeral string [ELMS, 1996][PROCTER & ELMS, 1998];
- The hybrid strategy, or over-segmentation, where a dissection algorithm is applied to the numeral string with the objective of segmenting it into many primitive segments. An optimization algorithm is then used to find the most

promising segmentation. Normally, a split-and-merge scheme based on graphs [NISHIDA & MORI, 1994] [LETHELIER et al., 1995] [HA et al., 1998] [OLIVEIRA et al., 2000] or dynamic programming [GADER et al., 1997] has been used to implement this strategy.

Another taxonomy divides the analytic methods into two approaches: a) segmentation-based methods which use an explicit segmentation of the string into numerals without the aid of a numeral recognizer; and b) segmentation-free methods which may use implicit or explicit segmentation. In both, a character recognizer is used to aid the segmentation process. Notice that 'segmentation-free' does not mean that no segmentation process is involved. It means rather that the segmentation process is performed with the aid of a numeral recognizer. The recognition can be performed simultaneously with the segmentation process (implicit segmentation), or afterwards, in order to search for the best way to assemble primitive segments to form the string (explicit segmentation).

2.2.1 Some important contributions

In [WANG et al., 1998], the GSC (Gradient, Structural and Concavity) digit recognizer proposed in [FAVATA et al., 1994] is adapted to recognize pairs of touching digits (00 to 99). The grid used for feature extraction is larger than that used for isolated digit recognition (4x6 instead of 4x4), and the pattern width is divided into three zones. Left and right zones are considered more useful than the central zone, since they better represent the numerals involved. For this purpose, the contribution of different zones is weighted. The recognition rate on a set of 523 touching-digit pairs extracted from the US postal address database is 86.8%.

In [SHI et al., 1997], the authors proposed a system for segmentation and recognition of totally unconstrained handwritten numeral strings. This segmentation-based approach consists of three modules: preprocessing, segmentation and recognition. In the preprocessing module, the objective is to segment the numeral string image into blobs containing one or more numerals. Initially, broken strokes caused by binarization are grouped by means of a filter, which blurs the input image of the numeral string to obtain a masked image. For that, each vertical foreground run-length is increased by one pixel. However, this filter is not sufficient to group broken parts caused by two-stroke

digits, such as 4 or 5. For this purpose, a procedure is proposed to group "good small components" to their closest neighbors using heuristic rules based on distance measures between strokes.

In this procedure, a first set of heuristic rules is used to remove noise. Small components far away from the others and long slim components are candidates for noise. Afterwards, a second set of heuristics is derived from the base line of the numeral string and its skew information, plus the average height and average width of the resulting components. This heuristic set is used to classify the resulting components into “good big”, “good small’ and “bad small components” from noise candidates.

The components classified as good small components are grouped to their neighbors in a blob. This is done by measuring the distances from the center horizontal position of the good small components to their left neighbors’ right horizontal position, and to their right neighbors’ left horizontal position (see Figure 2-1).

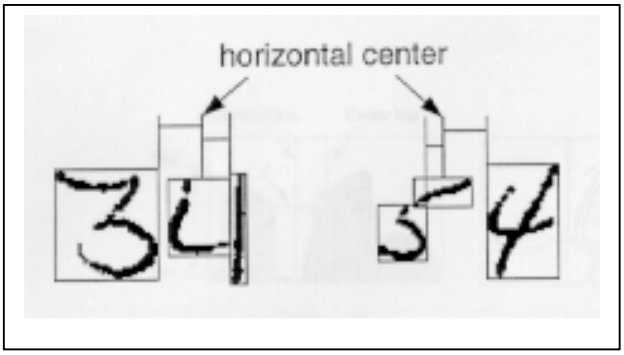


Figure 2-1 Grouping of broken parts (adapted from [SHI et al., 1997])

The number of numerals in the blob is estimated using a horizontal intersection method. Then, blobs with estimated number of numerals greater than one are sent to the segmentation module. The blob contour is used in this segmentation process. The objective is to detect significant right turns in the blob contour that are considered as touching points. For this purpose, the authors first compute vectors *P_in* leading into a contour point *P* from its several previous neighboring contour points, and *P_out* going out of *P* to its next several contour points. These vectors are normalized and placed in a Cartesian coordinate system with *P_in* along the *x*-axis. A significant right turn satisfies the following conditions:

$$x_1 y_2 - x_2 y_1 < 0 \qquad (2.2)$$

$$x_1 y_1 - x_2 y_2 < THR \tag{2.3}$$

where *THR* is experimentally determined (a number close to zero), (x_1, y_1) are the coordinates of *P_in*, and (x_2, y_2) are the coordinates of *P_out*. Equation 2.2 indicates that the turn is to the right, and equation 2.3 indicates that the turn is significant. *THR* ensures that the angle q made by *P_in* and *P_out* is close to 90°. Figure 2-2 shows that the distance between the thresholding line and the *y*-axis gives a threshold for determining a significant right turn.

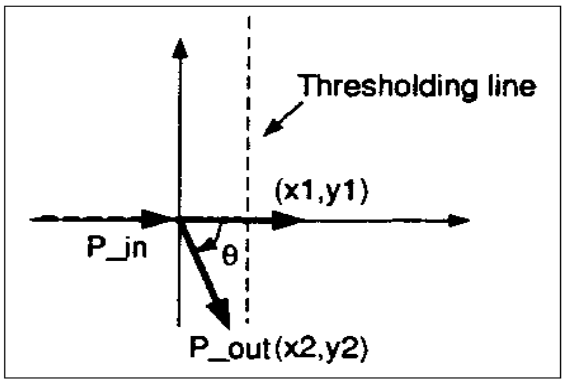


Figure 2-2 The distance between the thresholding line and the y-axis gives a threshold for determining a significant right turn (adapted from [SHI et al., 1997])

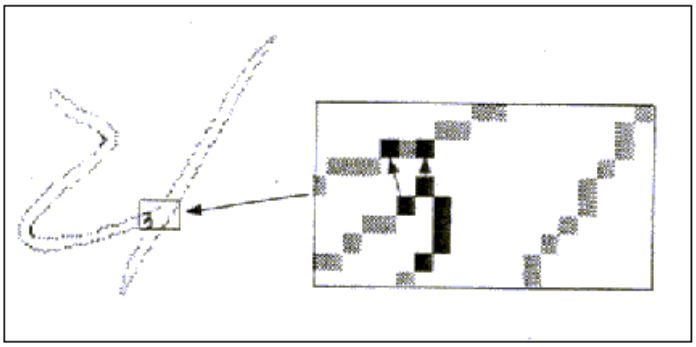


Figure 2-3 The significant right turning points and their opposite points divide the contour into contour pieces (adapted from [SHI et al., 1997])

The significant right turning points and theirs opposite contour points divide the contour into contour pieces, as shown in Figure 2-3. The authors assume that the number of numerals in the image input to the segmentation module is known. The segmentation result is obtained based on a histogram of vertical extents, which is calculated considering the vertical slant. A divide-and-conquer scheme is used to find *N* decision lines (segmentation lines) for *N*+1 numerals in the blob image. The intervals closest to the vertical bisecting line, and the minimal point within the valleys, are used

to define each decision line. Figure 2-4 shows a histogram of vertical extents used to find a decision line, and the segmentation result.

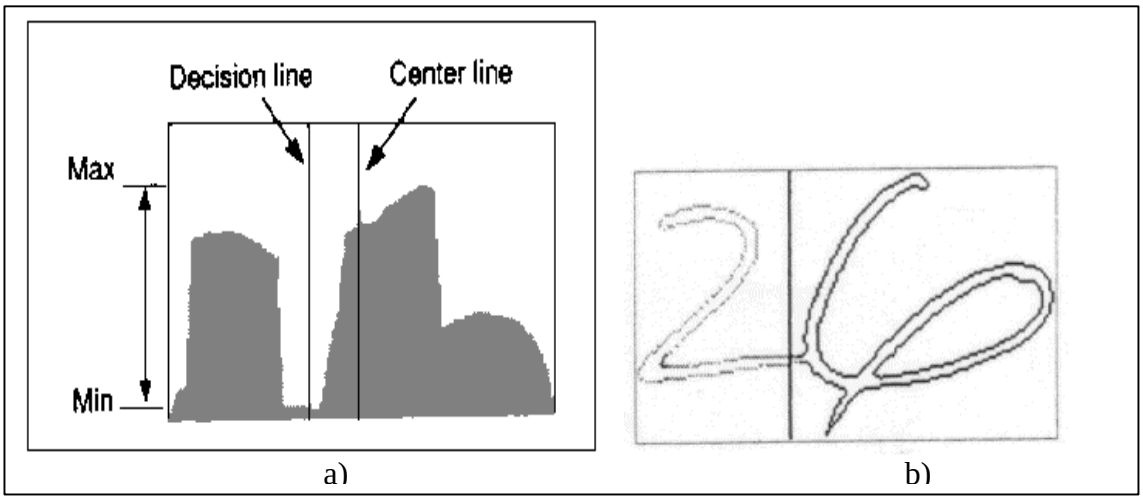


Figure 2-4 Segmentation proposed in [SHI et al., 1997]: a) Histogram of vertical extents used to find a decision line, and b) the segmentation result.

Finally, all the blobs with a single numeral are sent to the recognition module. The GSC isolated digit recognizer developed in CEDAR is used. Two experiments are reported. In the first, 495 zipcode images from CEDAR are used, for which an 85% field recognition rate and 97% segmentation rate are achieved. In the second, a set of 2,579 zipcode images is used. The recognition rate is 85.7%. For both, zero rejection is considered.

In [NISHIWAKI & YAMADA, 1998] we have a numeral string recognition method using character-touching-type verification strings. The authors propose a new segmentation method for recognition of numeral strings as a solution to mis-segmentation, which produces false digit candidates in conventional segmentation-based approaches. The method is based on checking touching type between a pair of character candidates. If the touching type between recognition results of the character candidates is impossible, they are rejected. Six touching types are used, which are shown in Figure 2-5.

The method consists of three processes: preprocessing, lattice generation and lattice estimation. Preprocessing erases granular noise and corrects the slant of the numeral string. The lattice generation process consists of detecting touching types in the numeral string image by comparing the length of a vertical black pixel run (*Vrun*) with that of the horizontally adjacent one. The leftmost and rightmost *Vruns* of each

component are also detected. A set of heuristics is used in order to detect each touching type in the image, resulting in a lattice of digit candidates. In this process, each digit candidate is still checked using the following heuristic rule: If the width is wider than a threshold value, the character candidate is eliminated from the lattice.

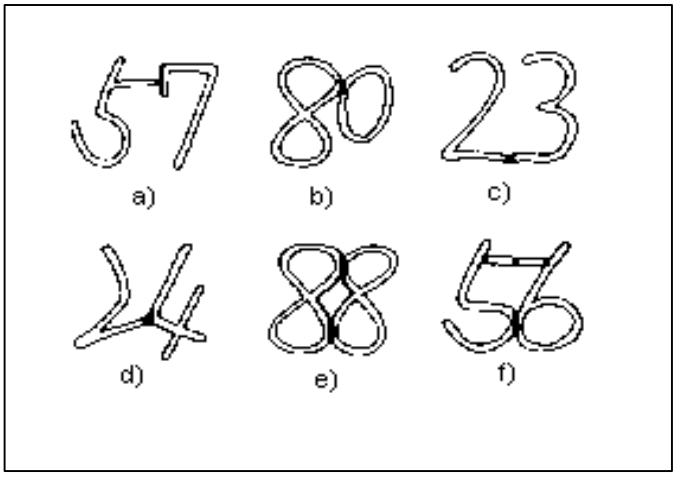


Figure 2-5 Touching types (adapted from [NISHIWAKI & YAMADA, 1998])

In the lattice estimation process, for each digit candidate a confidence value is calculated using an isolated character recognizer. Subsequently, all the verification units (pairs of digit codes and their touching type) are checked. If one of them is impossible, the confidence value of the digit code in it decreases by 20. Finally, the optimal path is selected from the lattice (see Figure 2-6). Experimental results are not reported.

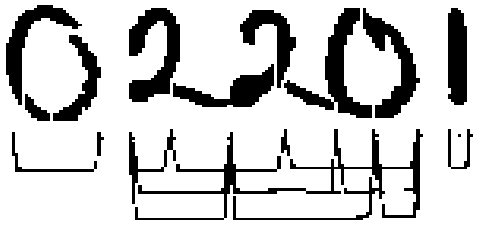


Figure 2-6 Example of a lattice and its possible segmentation paths (adapted from [NISHIWAKI & YAMADA, 1998])

In [PROCTER & ELMS, 1998], a segmentation-free method using implicit segmentation for the recognition of printed words, proposed in [ELMS, 1996], is applied to recognize handwritten numeral strings. Elms has used HMM-based recognition to avoid a prior segmentation of printed words into characters.

The feature extraction method proposed by Elms consists of a vertical shape profile. A shift-invariant shape feature is extracted from each column of pixels representing a pattern of bits. A shift-invariant Hamming distance measure is defined for shape quantization, and vector quantization is used to produce discrete observations from the feature space. The result is a codebook with 32 possible bit patterns. In addition, to avoid confusion caused by the character pairs (pb) and (dq) a COG (Center of Gravity) feature represents the relative center of pixel mass with respect to previous columns of pixels. During the recognition process, the input image is scanned from left to right, and for each vertical column a shape feature and a COG are calculated. The LBA [RABINER & JUANG, 1993] is used to match character HMMs against the input image.

In [PROCTER & ELMS, 1998], the same approach is used to recognize numeral strings of unknown length. The focus of their work is a study of the maximum number of levels in the LBA. The objective is to determine the point where exploration of further levels would not significantly improve system results. In a first attempt, the maximum number of levels for each field is defined as proportional to the width of the field in pixels, $L = x/x_0$, where x is the width of the observation sequence in pixels, and x_0 is a constant (between 10 and 27). The best performance is achieved with $x_0 = 22$, which is equivalent to approximately two levels per digit for the test data used. In another attempt, an adaptive level building is used. In this case, the exploration of further levels stops when the results are not significantly improved. This is done by examining the probabilities of the matches produced at each level of the LBA. The LBA is terminated when the probability of the best match at the current level (or next 2 or 3 levels) is lower than that of the previous level.

Experiments were conducted using 6,500 digits in 1,400 fields (length 2,3,4,5,6 and 10 digits) extracted from the NIST Special Database 1. In the first experiment ($x_0 = 22$), the field and isolated digit recognition rates are 74.7% and 93.25%, respectively. For the adaptive level building, the best results are 74.2% and 93.25% respectively.

Another segmentation-free method is described in [GADER et al., 1997]. This method uses an over-segmentation strategy. The first step consists of segmenting the handwritten numeral string into several primitive segments (digits or part of digits). Subsequently, a Kohonen SOFM (Self-Organizing Feature Map) plus an MLFN

(Multilayer Feedforward Neural Network) are used to assign low digit-class membership values to the primitives that represent parts of digits. Since the activation function at an SOFM node represents the distance between the weight vector at the node and an input pattern, the SOFM measures the typicality (or fuzzy set membership value). Thus, the SOFM output is used as the input to the MLFN, which assigns low confidence values to non-digits.

Finally, dynamic programming finds the best way to assemble the primitive segments to form a digit string, considering each potential number of digits (see Figure 2-7). Recognition rates are not reported.

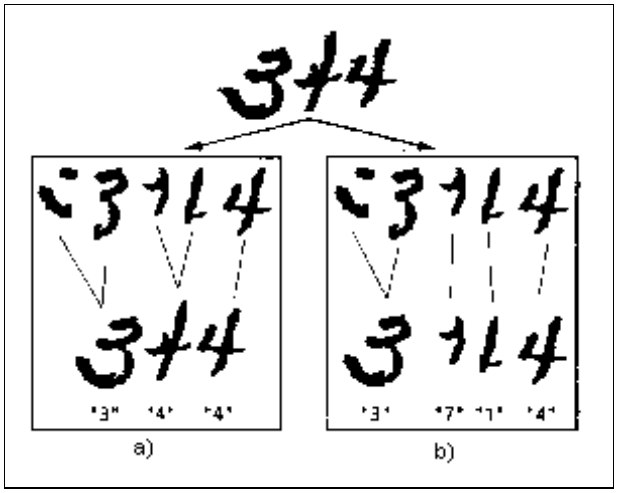


Figure 2-7 A numeral string over-segmented into digits or parts of digits, which are assembled to form a numeral string using dynamic programming (adapted from [GADER et al., 1997]); a) result considering the string length = 3; b) result considering the string length = 4.

In [HA et al., 1998], the authors combine segmentation-based and segmentation-free methods to construct an off-line handwritten numeral string recognition system. The architecture is shown in Figure 2-8. In the pre-segmentation module, the string image is divided into partial images (*PIs*). For this purpose, small components are eliminated and broken parts are grouped to their neighborhood, taking into account a set of heuristic rules. The resulting *PIs* are sent to the Digit Detection module, where they are classified as isolated digits or group of digits. The *PIs* classified as isolated digits are considered meaningful partial shapes (*PSs*).

On the other hand, the *PIs* classified as groups of digits are sent to the Segmentation-free module. In this module, a split-and-merge approach segments each

PI into partial shapes, which are merged into *meaningful PSs* , susceptible to represent individual numerals. *PI* segmentation consists of thinning and removing singular points such as end-point, T-joint or crossing point. In this process, each resulting connected component is labeled and expanded within the borders of the original *PI*, generating new *PSs*. In order to avoid exponential complexity during the merge of these *PSs* into *meaningful PSs* some spatial constraints are used.

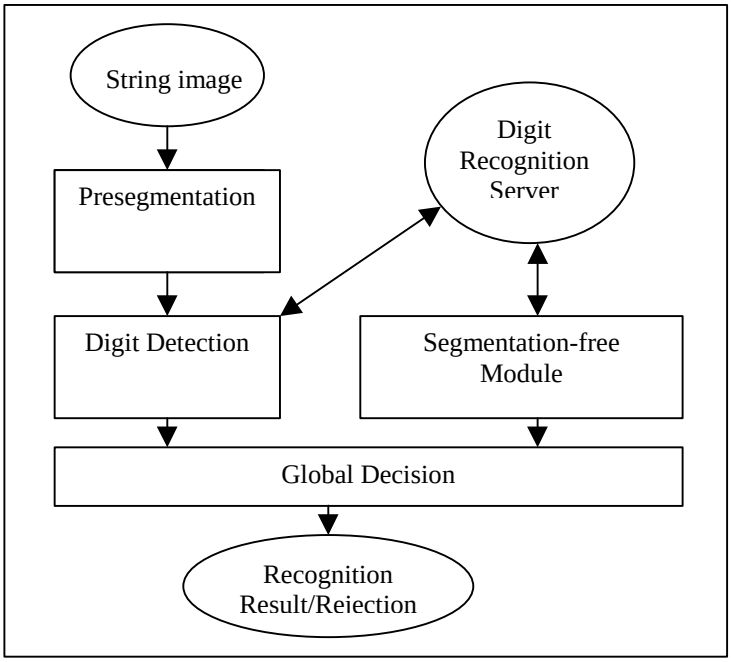


Figure 2-8 System architecture proposed in [HA et al., 1998]

These spatial constraints have been proposed in a similar segmentation-free approach based on a split-and-merge method in [NISHIDA & MORI, 1994], as follows: a) the characters are aligned horizontally; b) the segments are indexed from left to right; c) a character should not be very wide compared to its height; and d) a character should not be very short compared to its height. Moreover, a *PS* must satisfy additional size and shape constraints to be considered meaningful. Afterwards, each *meaningful PS* is sent to a digit recognizer. The class, score and *PS* coordinates are added to an attribute table. An additional score is computed, based on the digit score and how well the *PS* is embedded in the *PI*. The objective is to evaluate the quality of the *PSs*.

All possible *PS* combinations are represented by means of a directed and weighted graph generated from the attribute table. The cost of each node is derived from the *PS* scores. A best-first graph search is used to find the path with the lowest

cost. Some heuristics are used to treat additional noisy information in the graph (ligatures between digits, hyphen signs, underscores, *etc*.). Finally, the best path is sent to a *global decision* module, where the recognition result is accepted or rejected. The lowest score among all scores for the whole string (there is one for each digit) is assigned to the string.

An experiment using 4,925 samples of numeral strings (2, 3, 4, 5 and 6 digits in length) from the NIST database is reported. The global recognition and rejection rates are 92.7% and 0%, respectively. The recognition rates for numeral strings composed of 2, 3, 4, 5, and 6 digits are 96.2%, 92.7%, 93.2%, 91.1% and 90.3% respectively. Another experiment uses 495 numeral strings (5 and 9 digit lengths) from the CEDAR database. A recognition rate of 83.6% is reported, with zero rejection.

An integrated segmentation and recognition method using a cascade neural network composed of 747 nodes and 33,400 connections is proposed in [LEE & KIM, 1999]. This neural network is used to train the spatial dependencies in connected handwritten numerals. The general system architecture is shown in Figure 2-9. The string is preprocessed in order to reduce slant and size variations. Then, as a sliding input window scans the string with a step of two pixels (see Figure 2-10), the string is presegmented and the cascade neural network determines whether or not the segmentation is correct. If the segmentation is correct, the network classifies what is centered in the input window. For training the system, the content of the input window is paired with a target output vector representing what is centered in the input window at each stopping point.

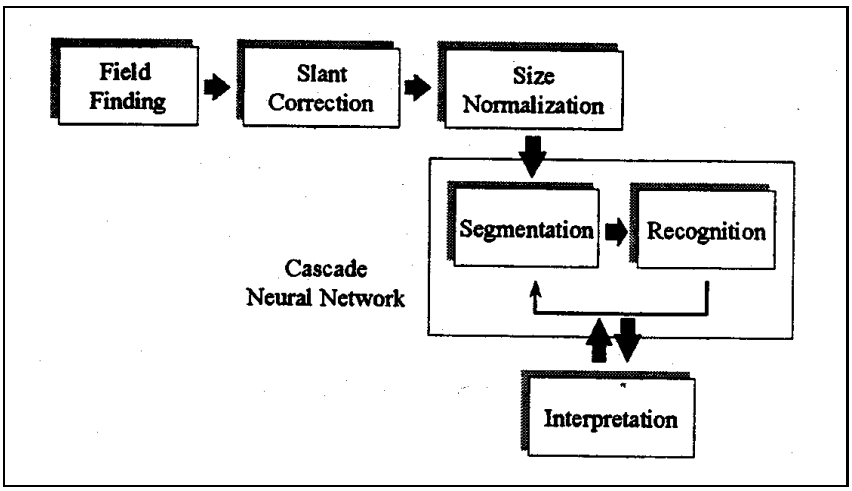


Figure 2-9 System architecture proposed in [LEE & KIM, 1999]

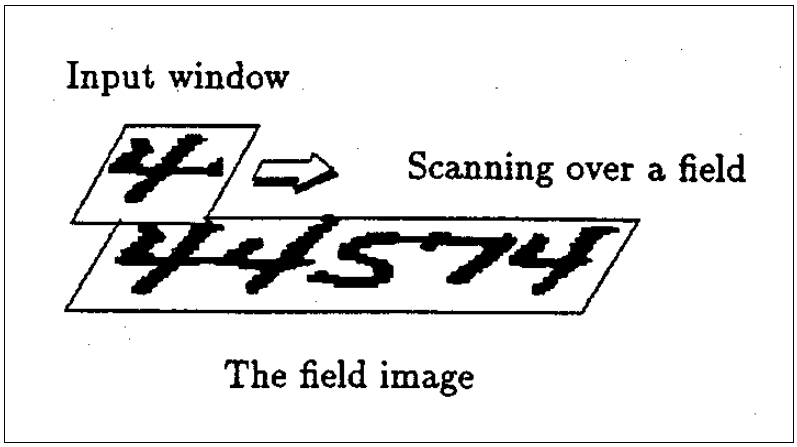


Figure 2-10 Sliding window proposed in [LEE & KIM, 1999]

During recognition, the activation value of each output unit represents what is centered in the input window. When the input window is centered in a character, the output unit corresponding to that character has a high activation value and others have a low value. In contrast, when the input window is centered between characters, a noncharacter class unit has a high activation value. A set of 5,000 numeral strings extracted from NIST database and equally distributed into 5 string classes (2,3,4,5 and 6-digit string) is used for testing. The recognition rates for numeral strings composed of 2, 3, 4, 5 and 6 digits are 95.23%, 88.01%, 80.69%, 78.61% and 70.49% respectively. The level rejection rate is not reported.

In [MATAN et al., 1992], an extended backpropagation learning neural network as a space displacement neural network (SDNN) is proposed. The authors use the SDNN in order to avoid replication of the recognizer at all possible locations across the input string. In their method, a size-normalized image is passed to the recognition system to generate a feature map. Then, the feature map is used to segment the string. A set of 3,000 numeral strings composed of 5 digits extracted from the NIST database is used for testing. The string recognition and error rates considering 0% string rejection rate are 66.3% and 33.7%, respectively.

In [MATIN et al., 1993], the authors propose the exhaustive and saccadic scan methods for integrating segmentation and recognition of handwritten strings. In the first scan method, a backpropagation learning neural network exhaustively scans a numeral string, and it is trained to recognize whether its input window is centered over a single digit or between digits. When its input window is centered on a digit, it is classified. The weakness of this method is that it generates too many candidate segments to be

efficient. In the saccadic scan method, the neural network is trained not only to recognize whether a character is centered on its input window, but also to compute ballistic “eye” movements that enable the input window jump from one digit to the next. A set of 5,000 numeral strings extracted from the NIST database and equally distributed into 5 string classes (2-,3-,4-,5- and 6-digit string) is used for testing. No recognition rate has been reported. Thus, we calculated it from the reported reject and error rates.Using the exhaustive scan method, the string rejection rates for numeral strings composed of 2, 3, 4, 5 and 6 digits considering a 1.0% string error rate are 4.8%, 11.1%, 19.1%, 23.4% and 35.7% respectively. The corresponding recognition rates are 94.2%, 87.9%, 79.9%, 75.6% and 63.3% respectively. For the saccadic scan method, the string rejection rates considering a 1.0% string error rate are 6,4%, 12,7%, 19,5%, 23,2% and 26,8% respectively. The corresponding string recognition rates are 92.6%, 86.3%, 79.5%, 75.8% and 72.2% respectively.

Keeler and Rumelhart use a neural network and the backpropagation algorithm to propose a system that simultaneously segments and recognizes connected characters [KEELER & RUMELHART, 1992]. Their Self-Organizing Integrated Segmentation and Recognition (SOISR) system takes position-independent information as targets and self-organizes the activities of the units in a competitive way to infer the positional information. A set of 5,000 numeral strings extracted from the NIST database and equally distributed into 5 string classes (2-,3-,4-,5- and 6-digit string) is used for testing. No recognition rate has been reported. Thus, we calculated it from the reported reject and error rates. The string rejection rates for numeral strings composed of 2, 3, 4, 5 and 6 digits considering a 1.0% string error rate are 12.0%, 15.0%, 23.0%, 28.0% and 37.0% respectively. The corresponding string recognition rates are 87.0%, 84.0%, 76.0%, 71.0% and 62.0% respectively.

In [FUJISAWA & NAKANO, 1992], the authors propose a region-based segmentation method for character segmentation and recognition, which takes into account the stroke shapes of touching patterns. The stroke shapes are analyzed in the case of touching characters. This system first extracts the connected components (CCs) from a numeral string. These CCs are analyzed in terms of spatial interrelations. They can be grouped into meaningful character patterns or separated by means of a method for finding the touching position. Multiple hypotheses and verification based on digit

recognition are used to deal with ambiguities. A set of 5,000 numeral strings extracted from the NIST database and equally distributed into 5 string classes (2-,3-,4-,5- and 6-digit string) is used for testing. The string recognition rate for numeral strings composed of 2, 3, 4, 5 and 6 digits are 89.79%, 84.64%, 80.63%, 76.05% and 74.54% respectively. The level rejection rate is not reported.

A method to recognize handwritten numeral strings without an explicit segmentation is presented in [YOON et al., 2000]. The authors adopt the concept of continuation and introduce the technique of subgraph matching to predefined prototypes. Continuation is a fundamentaly intuitive property of perceptual grouping in line segregation. A natural line, including not only the straight line but also the curved line, possesses the property of continuation if it has a smooth property of orientation over the line. This approach makes the segmentation of strings into digits unnecessary, since it does not guess the possible break positions and also because it recognizes a digit even if additional strokes are attached to it. A set of 100 numeral strings extracted from the NIST database and distributed into 6 string classes: 2_digit (24 samples), 3_digit (21 samples), 4_digit (21 samples), 5_digit (17 samples), 6-digit (12 samples) and 10_digit string (5 samples) is used for testing. The string recognition rates for numeral strings composed of 2, 3, 4, 5, 6 and 10 digits are 92.0%, 100.0%, 95.0%, 94.0%, 92.0%, and 100.0% respectively. The level rejection rate is not available.

2.3. Discussion

In the above sections, we have presented a brief review of handwritten isolated numeral recognition, since a string is usually recognized from the recognition of its individual numerals. In this review, we have included some investigations which have contributed to increasing recognition rates in recent years, such as:

.

- The combination of different feature types: in particular the combination of structural and statistical features, which has ensured an accurate character description, given their complementary properties;
- The combination of multiple classifiers, which has been important to allow the use of several feature extractors since different types of features may need different types of classifiers;

- The use of a verification module as a separate part of the recognition process, which permits the use of different knowledge levels to verify the recognition results, such as an additional set of features (low-level), or rules specified by the user (high-level).

Even with recognition rates close to 99% for handwritten isolated numerals, we can say that there is still a gap between human and machine performances. This gap is even greater when we consider the recognition of handwritten numeral strings, and is caused by the string difficulties described previously in this chapter. However, there have been important contributions on handwritten numeral string recognition. For example, some work has considered a holistic approach to string recognition, since this has been successfully used to recognize cursive words from a dictionary. The advantage of this approach is that the segmentation process is completely avoided, but the disadvantage is that it is unsuitable for handwritten strings of unknown length. It is nevertheless useful in dealing with specific problems, such as the recognition of touching-digit pairs, as proposed in [WANG et al., 1998].

The more conventional analytical methods are strongly based on heuristics. These methods attempt to segment the numeral string into individual numerals (segmentation-based methods). Thus, broken numerals represent a significant problem, since their parts need to be grouped to form a meaningful component that is able to represent an individual numeral. Usually, a set of heuristic rules is used to group parts of broken numerals. These rules normally take into account information like height, width and position of adjacent connected components, plus the distance between them. Examples of the effort required to group broken parts of numerals can be found in [SHI et al., 1997] and [HA et al., 1998].

Another difficult task for segmentation-based methods consists of segmenting touching numerals. Many studies have addressed just this problem. In [KIMURA & SHRIDHAR, 1992], upper and lower profiles plus a set of heuristics are used to determine the segmentation points. A segmentation path is constructed upward from the highest point on the lower profile of a numeral, or downward from the lowest point in the upper profile. In another approach, a contour analysis of connected numerals is performed [WESTALL & NARASIMHA, 1993]. Vertically-oriented edges are derived

from adjacent strokes, and these are used as vertices of a graph. These vertices are considered potential points of segmentation. Recently, in [YU & YAN, 1998], the distribution of structural features is used to determine the touching region between two numerals. Candidate touching points are selected from this region, using geometrical information. The recognition of the left or right lateral numeral is used to correct the position of the candidate touching point.

In general, these studies have revealed that segmenting touching characters without the aid of a recognizer is often unreliable. In addition, the use of heuristics to drive a blind search for digits in strings reduces the accuracy of the segmentation-based methods.

The segmentation-free methods have demonstrated their advantages in dealing with broken and touching numerals. An example is shown in [GADER et al., 1997], where an over-segmentation strategy is used to segment the string into primitive segments, which are classified as digits or parts of digits. Parts of digits must receive low recognition rates. A clear drawback of this strategy is the time it takes to merge primitive segments to form a string. Some studies have proposed the use of spatial constraints to avoid exponential complexity, as in [NISHIDA & MORI, 1994] and [HA et al., 1998]. However, these spatial constraints are again usually based on heuristics. Additional problems can be caused by misclassification of digit parts as digits. In Figure 2-7(b), we can see this problem, which is solved by considering the string length as known.

An alternative aimed at avoiding the over-segmentation problems has been the use of an implicit segmentation strategy to integrate segmentation and recognition. However, the cost to integrate segmentation into the recognition process is some loss of recognition performance. The work described in [PROCTER & ELMS, 1998] represents an example of this approach. The authors have adapted an HMM-based system, originally created to recognize printed words, for handwritten numeral strings. Unfortunately, the recognition results are not too good, because the proposed column shape features are not appropriate for modeling handwritten text. Moreover, the recognition performance of touching digits is not reported, and the overlapping between adjacent digits, which may represent additional problems for their feature extraction method, is not considered. Other examples of implicit segmentation-based methods are

found in [MATAN et al., 1992] [MATIN et al., 1993], [KEELER & RUMELHART, 1992] and [LEE & KIM, 1999], in which neural networks have shown to constitute a suitable framework for integrating the segmentation and recognition processes. However, the problem with these methods is the size definition of the sliding window used as input for the network, and the corresponding scan rate. Another common weakness is that they generate too many candidates segments when an exhaustive scan method is used.

In summary, we may conclude that an implicit segmentation-based method is a promising way to deal with the string difficulties described in Chapter 1, since the prior segmentation of strings into digits is avoided. Moreover, the method can be developed without the use of heuristics, which usually reduce the accuracy of the method. However, it is necessary to take into account some loss in terms of recognition performance which may be brought about by integrating segmentation into the recognition process.

2.4. Summary

In this chapter, we have presented the main topics related to the proposed method: handwritten isolated numeral recognition, and handwritten numeral string recognition. The main investigations that have permitted improvements to be made in recognition performance of isolated handwritten numerals have been presented. Some recent work has been briefly described in terms of features, types of classifiers, test databases and results. Moreover, different approaches for recognizing numeral strings have been presented and discussed. In the next chapter, the main concepts related to the background theory necessary to describe the proposed method are presented.

3. Background Theory

In this chapter, a brief introduction to the background theory relevant to the method for handwritten numeral string recognition proposed in this thesis is presented. The Hidden Markov Model (HMM), which is used in both SCB and Verification stages of the proposed method, is described in Section 3.1, including the main HMM concepts, types of HMMs and the algorithms required to implement this statistical technique for modeling real problems. Some aspects of observation sequences (discrete, continuous and semi-continuous) are discussed. Finally, we present a scheme to optimize the number of states of an HMM. A vector quantization process, which is used for mapping feature vectors calculated from a digit or string image onto discrete observations, is described in Section 3.2, as is the K-means algorithm. Finally, in Section 3.3, a brief review of Bayesian theory is provided. The theory is used in the proposed method to implement a classifier to predict the string length (number of digits) from its bounding box width (in columns). This length predictor is used to deal with strings of unknown length in the last experiments of the proposed evaluation protocol.

3.1. Hidden Markov Model (HMM)

The original motivation for using HMMs for handwritten numeral string recognition was based on their successful application to the recognition of spoken strings in the field of Speech Recognition [RABINER, 1989]. As in handwritten numeral strings, the number of words in spoken strings, and their boundaries, are unknown. Moreover, a priori segmentation of spoken strings into words is often unreliable because of sound coarticulation. In [RABINER & JUANG, 1993], a system for recognizing spoken digit strings is described, and various algorithms for optimal matching of single digit HMMs against an unknown spoken string are discussed. More recently, in [BOSE & KUO, 1994] and [ELMS et al., 1998], the authors have shown the benefits of applying HMMs to represent and recognize printed text in the field of off-line recognition. This research has proved that HMM is particularly well suited to

performing segmentation during recognition. In this direction, the proposed method uses HMMs to implement an implicit segmentation to deal with the string difficulties described in Chapter 1.

A detailed review of this kind of statistical modeling and its application to the machine recognition of speech can be found in [RABINER, 1989]. In [PORITZ, 1988] the basic idea is introduced by elementary examples, and the Baum-Welch algorithm for maximum likelihood estimation of the model parameters is discussed from both an intuitive and a formal point of view.

The Markov chain is a method for modeling a signal as a sequence of observable outputs produced by some process called the *source*, or the *Markov Source*. In such a source, the symbols currently produced are dependent only on a fixed number of symbols which had been produced preceding the current output. The number of preceding outputs taken into account for the next symbol to be produced defines the order of a Markov Model. First- and second-order Markov Models have been used for most applications, since the complexity of the model grows exponentially with the order. This work focuses on first-order Markov Models.

The HMM has the same structure as a Markov chain, but the difference is that each state of a Markov chain is a single observation symbol corresponding to an observable physical event. In an HMM, each state is a probability over all symbols. Thus, in a Markov chain, it is possible, given a sequence of symbols produced by a model, to compute the sequence of states that produced it. However, in most real problems, each state of a model cannot be defined as a single symbol, because more than one symbol can be observed per state. In order to extend the Markov chain to deal with these problems, the states in an HMM model are regarded as “hidden”, *i.e.* each state is a probability distribution over all symbols. Thus, HMMs cannot provide the exact sequence of states that produced a given sequence of symbols generated by a model, but it is possible to compute the sequence of states with the highest probability of having produced the sequence of symbols observed.

The following notation describes a complete parameter set of a first-order discrete HMM:

$$l = (N, M, T, A, B, p) \tag{3.1}$$

where:

N: number of distinct states in the model. The set of states can be written as:

$$S = \{s_1, s_2, ..., s_N\}, \tag{3.2}$$

and q_t denotes the state at time t;

M: number of distinct observation symbols per state (size of alphabet), which represent the physical output of the model. The set of the individual symbols can be denoted by:

$$V = \{v_1, v_2 ..., v_M\}. \tag{3.3}$$

T: length of the observation sequence. The observation sequence can be denoted by:

$$O = \{o_1, o_2, ..., o_T\}, \tag{3.4}$$

and o_t denotes the observation at time *t*;

A: state transition probability distribution. $A = \{a_{ij}\}$, where:

$$a_{ij} = P[q_{t+1} = s_j \mid q_t = s_i], \qquad 1 \le i, j \le N, \tag{3.5}$$

i.e, the probability of the state s_j at time $(t+1)$ given state s_i at time *t*. In order to obey the standard stochastic constraints, the elements in *A* must present the following properties:

$$a_{ij} \ge 0, \qquad \forall j, i \tag{3.6}$$

$$\sum_{j=1}^{N} a_{ij} = 1, \qquad \forall i\,. \tag{3.7}$$

B: observation symbol probability distribution in state j. $B = \{b_j(k)\}$, where

$$b_j(k) = P[o_t = v_k \mid q_t = s_j], \qquad 1 \le k \le M,\ 1 \le j \le N\,. \tag{3.8}$$

The probability of the symbol v_k being observed at time *t*, given state s_j at time *t*, $b_j(k)$ must have the following properties:

$$b_j(k) \ge 0 \tag{3.9}$$

$$\sum_{k=1}^{M} b_j(k) = 1\,. \tag{3.10}$$

p: initial state distribution $p = \{p_i\}$, where:

$$p_i = P[q_1 = s_i], \qquad 1 \le i \le N, \tag{3.11}$$

the probability that the initial state equals s_i. Similarly, this probability must be non-negative and:

$$\sum_{i=1}^{N} p_i = 1. \tag{3.12}$$

3.1.1. Types of HMMs

There are two important types of HMMs: ergodic and left-right or Bakis model [RABINER, 1989]. The ergodic model is a specific case of a fully-connected model when all a_{ij} are positive. In this type of model, the states are interconnected in such a way that any state can be reached from any other state. Figure 3-1(a) shows a 4-state ergodic HMM model.

The left-right model presents an important kind of state interconnection for text recognition modeling which has the property:

$$a_{ij} = 0, \qquad j < i. \tag{3.13}$$

This property means that no transitions are allowed to those states whose indices are lower than the current state, which is of interest for modeling signals changing over time. Figure 3-1(b) presents a 4-state left-right model. Since the state sequence must begin in state 1 and must end in state N, the initial state probabilities have the following property: $p_i = 0$ when $i \neq 1$, and $p_i = 1$ when $i = 1$.

Often, with left-right models, additional constraints are used, such as:

$$a_{ij} = 0, \qquad j > i + \Delta, \tag{3.14}$$

in order to avoid great changes in state indices. Δ is a value used as a limit for jumps. For example, in Figure 3-1(b), Δ is 2; then, no jumps with more than 2 states are allowed.

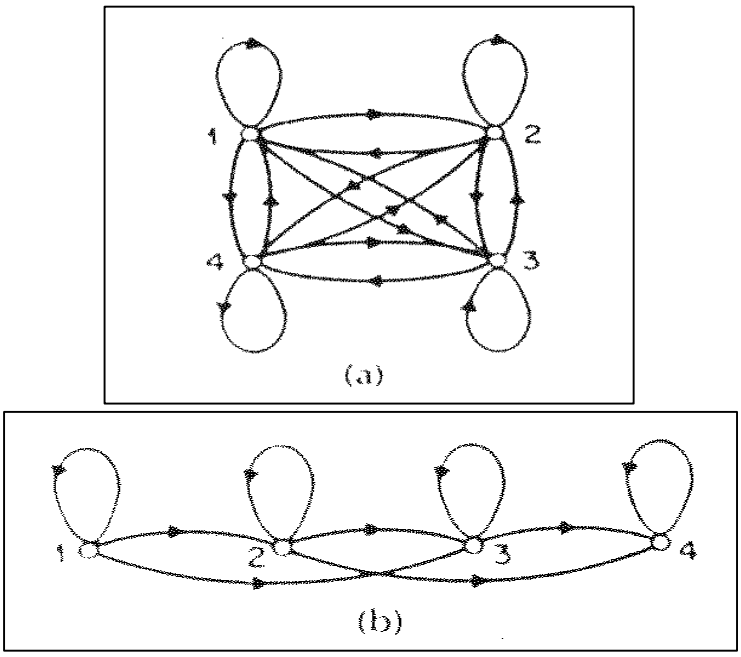


Figure 3-1 Types of HMM; a) a 4-state ergodic HMM model; b) 4-state left right model

3.1.2. Discrete, continuous and semi-continuous observation densities

As mentioned previously, each HMM state has an observation symbol probability distribution (parameter B), which describes the probability of the symbol v_k being observed in this state. In a discrete HMM model, the probability of observing a symbol v_k while in state s_j, denoted as $b_j(k)$, is defined from a distribution computed over the set of all possible symbols in the system. This distribution is non-parametric and quantified, *i.e.* it is not needed *a priori* knowledge about the distribution form to model a signal, and each observation can take only one discrete value from a finite set. Then, it is necessary only to have enough training data to model a signal. However, for most applications, the observations are continuous signals. In these cases, a discrete HMM can be used only after a quantization process of the signal in order to create a codebook. The cost is that the quantization process usually presents a quantization distortion, which may degrade performance significantly. Moreover, to add a new class to the system, the reconstruction of the codebook is necessary, and consequently re-training of all system models.

The use of continuous HMMs makes it possible to avoid quantization distortion and retraining of the system, since there is no codebook. However, there are some

costs: a continuous probability density function must be defined *a priori*, and more training data is required for an accurate HMM parameter estimation. A mixture density function, weighted using the sum of a number of parametric distributions, has been used to find the best way to reflect the distribution of observations.

$$b_j(\underline{x}) = \sum_{m=1}^{M} c_{jm} b_{jm}(\underline{x}). \tag{3.15}$$

Here c_{jm} is the weight for the *m-th* mixture in state s_j, and *M* is the number of mixture components.

Semi-continuous HMMs, or SCHMMs, can be used to avoid distortions caused by quantization of continuous signals to model them with discrete HMMs, as well as to reduce the amount of data and computational complexity to train continuous HMMs. Also called tied mixtures, the idea is that similarities may exist in the data between observations that do not represent the same source. This means that clustering the data in an unsupervised way should create clusters with crossing class boundaries. We can represent any state distribution as weighted combination of Gaussian prototypes known as semi-continuous densities.

In SCHMMs, the vector quantization (VQ) codebook used to model continuous signal with discrete HMMs is represented by a set of continuous probability density functions, whose distributions are overlapped. In this codebook, each codeword can be represented by a continuous probability density function. Then, the VQ operation produces values of continuous probability density functions $f(x \mid v_j)$ for all codewords v_j.

The semi-continuous output probability can be considered as a mixture probability density function, with the *K* codewords in the codebook being mixed using the *B* parameter of the HMM model as weighting coefficients, as below:

$$b_j(\underline{x}) = \sum_{k=1}^{K} f(\underline{x} \mid v_k) b_j(k). \tag{3.16}$$

The problem with this method is that it focuses on clustering similar observations, and not observations which provide discriminating information.

In [HUANG et al., 1993], an interesting comparative study of these types of observation densities is carried out. Given the advantages and disadvantages of each type of observation density, we decided to implement a discrete HMM to be sure that in both stages of the proposed method, SCB and Verification, there will be enough data for training. The problem is not to have enough samples of isolated digits in the NIST database, but enough samples to model specific handwriting knowledge related to the interaction between adjacent digits in strings. In addition, it is important to ensure portability to the proposed method.

3.1.3. Training HMMs

There are different ways to train an HMM. A detailed description of the training criteria can be found in [RABINER, 1989]. The most common criterion is called Maximum Likelihood (ML). In this criterion, given a training database composed of sequence of observations, the HMM parameters are first initialized and then iteratively re-estimated such that the likelihood of the model produced by the training sequences increases. The training process stops when the likelihood reaches a maximum value. The ML estimators focus on maximizing the likelihood of an individual model producing observations from the source that it is modeling, but it does not take into account the likelihood of competing models producing these same observations. This may produce suboptimal decision boundaries.

An alternative is to use the Maximum Mutual Information (MMI) criterion. In this case, a set of models is trained to maximize the ability of each model to discriminate between observation sequences generated by itself and those generated by the other models. This criterion is used to distinguish the correct model from all the other models on the training sequence. However, an analytical solution to this problem is not feasible.

A third option is the Minimum Discrimination Information (MDI) criterion, which is used when the signal to be modeled was not necessarily generated by a Markov source. This criterion minimizes the cross-entropy between the set of valid signal probability densities and the set of HMM probability densities. Unfortunately, obtaining such a minimum is highly non-trivial.

In this work, we use the Baum-Welch (BW) algorithm to train numeral models. This algorithm is based on ML criterion. The first step consists of calculating $P(O \mid l)$, i.e., the probability of the observation sequence O, given the model l.

The efficient solution to calculate $P(O \mid l)$ is called a forward procedure. Consider the forward variable, $a_t(i)$, defined as:

$$a_t(i) = P(o_1, o_2, ..., o_t, q_t = s_i \mid l), \tag{3.17}$$

which is the probability of the partial observation sequence o_1, o_2,…o_t , and state s_i at time *t*, given the model l. The algorithm for inducting $a_t(i)$ is described below:

Initialization:

$$a_1(i) = p_i b_i(o_1), \qquad 1 \le i \le N, \tag{3.18}$$

Induction:

$$a_{t+1}(j) = \left[\sum_{i=1}^{N} a_t(i) a_{ij} \right] b_j(o_{t+1}), \qquad \begin{matrix} 1 \le t \le T-1, \\ 1 \le j \le N, \end{matrix} \tag{3.19}$$

Termination:

$$P(O \mid l) = \sum_{i=1}^{N} a_T(i). \tag{3.20}$$

The objective of the BW algorithm is to adjust the model parameters $l = (A, B, p)$ to maximize $P(O \mid l)$. This is the most difficult problem in the HMM domain. The BW is an iterative algorithm based on the forward (previously defined) and backward probabilities. The backward variable $b_t(i)$, similar to forward variable $a_t(i)$, is defined as:

$$b_t(i) = P(o_{t+1}, o_{t+2}, ..., o_T \mid q_t = s_i, l), \tag{3.21}$$

which is the probability of the partial observation sequence from *t+1* to the end, given state s_i at time *t* and the model l. The algorithm for induction of $b_t(i)$ is described below:

Initialization:

$$b_T(i) = 1, \qquad 1 \le i \le N. \tag{3.22}$$

Induction:

$$b_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1})\, b_{t+1}(j), \qquad \begin{matrix} t = T-1, T-2, \ldots 1 \\ 1 \le i \le N. \end{matrix} \tag{3.23}$$

After defining the backward variable, it is possible to define the probability of being in state s_i at time *t*, and state s_j at time *t+1*, given the model and the observation sequence, which is denoted by:

$$x(i, j) = P(q_t = s_i, q_{t+1} = s_j \mid O, l), \tag{3.24}$$

and is calculated using:

$$x_t(i, j) = \frac{a_t(i)\, a_{ij}\, b_j(o_{t+1})\, b_{t+1}(j)}{P(O \mid l)}. \tag{3.25}$$

Now, consider $g_t(i)$ as the probability of being in state s_i at time *t*, given the observation sequence *O*, and the model l.

$$g_t(i) = P(q_t = s_i \mid O, l), \tag{3.26}$$

which can be calculated using the forward and backward variables, as:

$$g_t(i) = \frac{a_t(i)\, b_t(i)}{P(O \mid l)}. \tag{3.27}$$

Given that, the expected number of transitions from s_i is denoted by

$$\sum_{t=1}^{T-1} g_t(i), \tag{3.28}$$

and the expected number of transitions from s_i to s_j is denoted by

$$\sum_{t=1}^{T-1} x_t(i, j). \tag{3.29}$$

Then using the formulas above, it is possible to have a method to re-estimate the parameters p, *A* and *B* of an HMM, the set of formulas is constituted by:

1) expected frequency in state s_i at time $t = 1$

$$\overline{p}_i = \gamma_1(i). \tag{3.30}$$

2) transition coefficient = expected number of transitions from state s_i to s_j, divided by the expected number of transitions from state s_i.

$$\overline{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}. \tag{3.31}$$

3) observation symbol probability = expected number of times in state j, while observing symbol v_k, divided by the expected number of times in state j.

$$\overline{b}_j(k) = \frac{\sum_{t=1, s.t. o_t = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{3.32}$$

3.1.4. Scoring HMMs

The scoring of HMMs in the proposed method is done by two algorithms – Viterbi's Algorithm [RABINER & JUANG, 1993] and the Level Building Algorithm (LBA) [RABINER & JUANG, 1993][ELMS, 1996]. Viterbi's algorithm is used to implement the digit classifier used in the Verification stage. This algorithm is able to match a single model to an observed sequence of symbols. The LBA is used to implement the segmentation-recognition module based on implicit segmentation strategy in the String Context-Based stage. This algorithm is able to find the sequence of numeral HMMs that best matches an unknown numeral string.

Viterbi's algorithm

Viterbi's algorithm is a dynamic programming method to estimate the sequence of model states with highest probability of having produced the sequence of observations, *i.e.*, given the observation sequence $O = \{o_1, o_2, ..., o_T\}$, and the model l, this method finds a corresponding state sequence $Q = \{q_1, q_2, ..., q_T\}$, which best explains

the observations. It is used to match a single model to the observation sequence. Consider the following variables:

- $\delta_t(i)$: scores the likelihood of the observation sequence $o_1, o_2,\ldots,o_t$ having been produced by the most likely sequence of model states, which ends at state i at time t;
- $\psi_t(i)$: array used to trace the maximum likelihood path. It keeps a record of the states which maximized the likelihood from time 1 to t.

Viterbi's algorithm is described below:

Initialization:

$$\delta_1(i) = p_i b_i(o_1), \qquad 1 \le i \le N \tag{3.33}$$

$$\psi_1(i) = 0. \tag{3.34}$$

Recursion:

$$\delta_t(j) = \max_{1 \le i \le N} \left[\delta_{t-1}(i) a_{ij} \right] b_j(o_t), \qquad \begin{matrix} 2 \le t \le T, \\ 1 \le j \le N \end{matrix} \tag{3.35}$$

$$\psi_t(j) = \arg\max_{1 \le i \le N} \left[\delta_{t-1}(i) a_{ij} \right], \qquad \begin{matrix} 2 \le t \le T, \\ 1 \le j \le N. \end{matrix} \tag{3.36}$$

Termination:

$$P^* = \max_{1 \le i \le N} \left[\delta_T(i) \right], \tag{3.37}$$

$$q_T^* = \arg\max_{1 \le i \le N} \left[\delta_T(i) \right]. \tag{3.38}$$

Backtracking for state sequence:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \qquad t = T-1, T-2, \ldots, 1. \tag{3.39}$$

In the proposed method, Viterbi's algorithm is used for isolated digit classification in the Verification stage. The goal is to decide which numeral model an observed sequence was produced by. For this purpose, the sequence of symbols is presented to each model and its probability (P^*) of being produced by that model is evaluated. The model that has the greatest likelihood of producing this observation sequence defines the numeral class.

Level Building Algorithm

The Level Building Algorithm (LBA) [RABINER & JUANG, 1993][ELMS, 1996][PROCTER & ELMS, 1998] is used to find the sequence of numeral HMMs that best matches an unknown numeral string. Figure 3-2 shows a trellis structure that describes the LBA.

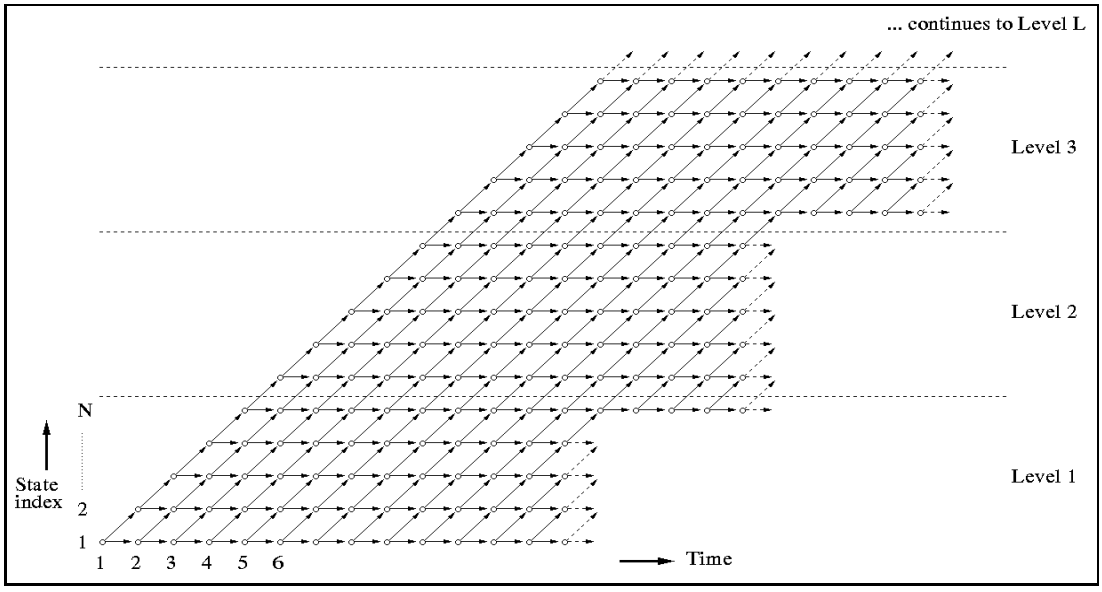


Figure 3-2 Trellis structure for LBA

For describing this algorithm let us consider $l = \{l_1, l_2, ..., l_m\}$ as a set of M left-to-right HMMs, each one composed of N states; $O = \{o_1, o_2, ..., o_T\}$ as an observation sequence of length T; t as a time function ($1 \le t \le T$); and L as the maximum number of levels, which must be large enough to enable the correct model sequence to be discovered. For example, 10 levels allow a total of 10 character models to be matched against the input sequence.

At the first level ($l = 1$), each model l_k ($1 \le k \le M$) is matched against the observation sequence from time $t = 1$.

Initialization:

$$\delta_1(1) = b_1^l(o_1) \tag{3.40}$$

$$\delta_1(i) = 0 \qquad 2 \le i \le N. \tag{3.41}$$

Recursion:

$$\delta_t(j) = \max_{1 \le i \le N} \left[\delta_{t-1}(i) a_{ij}^l \right] b_j^l(o_t) \qquad \begin{matrix} 2 \le t \le T \\ 1 \le j \le N. \end{matrix} \tag{3.42}$$

Termination:

$$P^*(l,t,\ l) = \delta_t(N), \qquad 2 \le t \le T \tag{3.43}$$

$$B^*(l,t,\ l) = -1. \tag{3.44}$$

A *level reduction* is performed at the end of the level. At this point the equations 3.45 and 3.46 are used to keep, respectively, the probability of the best match model and the corresponding model label. Equation 3.47 represents a back-pointer to the previous level.

$$P^{**}(l,t) = \max_l \left[P^*(l,t,\ l)\right]; \tag{3.45}$$

$$\Omega(l,t) = \arg\max_l \left[P^*(l,t,\ l)\right]; \tag{3.46}$$

$$B^{**}(l,t) = B^*\left(l,t,\arg\max_l \left[P^*(l,t,\ l)\right]\right) \tag{3.47}$$

For higher levels $(l \ge 2)$, since they pick up from the output of the previous level model, the initialization procedure must be:

$$\delta_1(1) = 0, \tag{3.48}$$

$$\delta_t(1) = \max\left[P^*(l-1,t-1), a_{11}^l\ \delta_{t-1}(1)\right] b_1^l(o_t), \qquad 2 \le t \le T. \tag{3.49}$$

Moreover, a back-pointer array must be added to keep track of the time on the previous level when the previous model match ended. This array must be initialized taking into account the following condition:

$$\alpha_t(1) = \begin{cases} t-1 & \text{if } P^*(l-1,t-1) > a_{11}^l\ \delta_{t-1}(1) \\ \alpha_{t-1}(1) & \text{otherwise} \end{cases} \tag{3.50}$$

During the *Recursion procedure* this back-pointer array is updated, as:

$$\alpha_t(j) = \arg\max_{1 \le i \le N}\left[\delta_{t-1}(i) a_{ij}^l\right]. \tag{3.51}$$

Finally, in the *Termination procedure* this back-pointer array is resumed in B^*, as:

$$B^*(l,t,\ l) = \alpha_t(N), \qquad 1 \le t \le T. \tag{3.52}$$

After constructing all levels, $P^{**}(L,T)$ represents the probability of the best match of L models to the observation sequence. In order to find the best match, the B^{**}

array can be used in a backtracking process. The best match string is the maximum of P^{**} over all levels.

During the construction of the proposed system, we fix parameter *L* since the string length is considered as known. Being a strong system constraint, this allows a fine tuning of important system parameters, such as the HMM topology in the LBA framework. In the last experiments we relax this constraint.

Another option of Dynamic Programming (DP) algorithm to match individual models against an unknown string is the Two-Level Dynamic Programming (Two-Level DP) Algorithm [RABINER, 1989]. The two-level DP algorithm is divided into two stages or levels: At the first level the algorithm matches individual models against an unknown string, while the second level corresponds to the search for the optimal concatenated string path. The main difference between the Level Building and the two-level DP algorithms is that partial digit decisions are used to reduce the search range in later stages of the LBA, while the two-level DP algorithm accumulates all the digit scores until the end of the entire process and then decides the digit string in a separate and independent second level DP calculation. Thus LBA involves significantly less computation than the two-level DP algorithm.

3.1.5. Defining the HMM length

[WANG, 1994] describes a method to define the possible number of states (*N*) of the HMMs taking into account durational statistics calculated from the training database. First the mean length m and the variance s^2 of all observation sequences in the training set are collected and they together define the possible *N* for each numeral HMM:

$$\frac{m(m-1)+s^2}{m-1+s^2} < N < m+1-\sqrt{2s^2+1} \tag{3.52}$$

Table 3-1 presents the range (minimum and maximum number of states) for each numeral model calculated on the training set (50,000 isolated digits – 5,000 per class). In addition, the mean length value is also calculated.

Table 3.1 Minimum, maximum and mean number of states by digit class

	Column based models			Row based models		
Numeral model	Min	Mean	Max	Min	Mean	Max
0	13	18	24	14	21	28
1	5	6	7	16	24	32
2	14	22	30	16	24	32
3	14	20	26	20	28	36
4	15	22	28	18	28	39
5	13	21	29	19	27	35
6	15	20	25	18	27	36
7	15	20	25	18	27	36
8	14	17	24	20	29	38
9	16	20	25	21	31	41

3.2. Vector quantization process

To use the HMMs described in the previous sections, we need to represent a string or digit image as a sequence of discrete observations. To this end, in the feature extraction of the proposed method each real-valued, continuous-amplitude feature vector needs to be quantized to one of a number of discrete symbols available in a previously computed codebook. To create this codebook, it is necessary to apply the concept of Vector Quantization (VQ) [LINDE et al., 1980] [MAKHOUL et al., 1985].

Let us assume that $x=[x_1, x_2, ..., x_N]$ is an N-dimensional vector whose components $\{x_k, 1 \le k \le N\}$ are real-valued, continuous-amplitude random variables. In vector quantization, the vector x is mapped onto another real-valued, discrete-amplitude, N dimensional vector y. It is used to say that x is quantized as y, and y is the quantized value of x. This can be denoted by:

$$y = q(x) \tag{3.53}$$

where $q(\ )$ is the quantization operator and y is the output vector corresponding to x. The value of y is a finite set of values $\{y_i, 1 \le i \le L\}$, where $y_i = [y_{i1}, y_{i2} ... y_{iN}]$. L is the codebook size (or number of levels), and $Y = \{y_i\}$ is the set of code vectors. To design the codebook, we partition the *N*-dimensional space of the random vector x into L regions or cells and associate with each cell C_i a vector y_i. The quantizer then assigns the code vector y_i if x is in C_i.

$$q(x) = y_i, \quad if \ \ x \in C_i \tag{3.54}$$

The mapping of x onto y results in a quantization error, and a distortion measure $d(x, y)$ can be defined between them, also known as dissimilarity. The most common distortion measure is the mean-square error, given by:

$$d(x, y) = \frac{1}{N} \sum_{k=1}^{N} (x_k - y_k)^2 \tag{3.55}$$

Quantization is optimal when distortion is minimized over all *L*-levels quantizers. There are two necessary conditions for optimality. The first condition is that the optimal quantizer is realized by using a minimum-distortion or nearest neighbor selection rule.

$$q(x) = y_i, \quad iff \ d(x, y_i) \le d(x, y_j), \ \ j \neq i, \ 1 \le j \le L \tag{3.56}$$

This means that the quantizer selects the code vector that results in the minimum distortion with respect to x. The second necessary condition for optimality is that each code vector y_i is chosen to minimize the average distortion in cell C_i. Let us consider $\{x(n), 1 \le n \le M\}$ as a set of training vectors, and M_i as a subset of those vectors in cell C_i. The average distortion D_i is then given by:

$$D_i = \frac{1}{M_i} \sum_{x \in C_i} d(x, y_i) \tag{3.57}$$

The vector that minimizes the average distortion in cell C_i is called the *centroid* of C_i, and it is denoted as:

$$y_i = cent(C_i) \tag{3.58}$$

One well-known method for codebook design is an iterative clustering algorithm known in the pattern recognition literature as the *K-means algorithm* [MAKHOUL et al., 1985], where $K = L$ (codebook size). The algorithm divides the set of training vectors $\{x(n)\}$ into *L* clusters C_i in such a way that the two necessary conditions for optimality

are satisfied. In the algorithm description, m is the iteration index and $C_i(m)$ is the i[th] cluster at iteration m, with $y_i(m)$ its centroid. The algorithm is as follows:

K-means algorithm

Inicialization Step:	Set $m = 0$. Choose a set of initial code vectors $y_i(0), 1 \le i \le L$.
Classification Step:	Classify the set of training vectors $\{x(n), 1 \le n \le M\}$ into the clusters C_i by the nearest neighbor rule. $x \in c_i(m), \quad iff\ d(x, y_i) \le d(x, y_j),\ all\ j \ne i,\ 1 \le j \le L$
Code Vector Updating Step	$m = m + 1$. Update the code vector of each cluster by computing the centroid of the corresponding training vectors in each cluster $y_i(m) = cent(C_i(m)),\ 1 \le i \le L$.
Termination Test Step	if the decrease in the overall distortion $D(m)$ at iteration m relative to $D(m-1)$ is below a certain threshold, stop; otherwise go to the *Classification Step*.

3.3. The Bayesian decision theory

Let us consider the problem of string length prediction, where it is necessary to decide the length (number of digits) of a numeral string in one of the M classes available in the set $w = \{w_1, w_2, ..., w_M\}$. In this set, class w_2 represents strings composed of 2 digits. Available information is the a priori probability of each class. For instance, the a priori probability of the class w_j may be computed by counting the total number, N, of all observations and the number of observations N_j which belongs to the class w_j. Then $P(w_j)$ can be defined as the relative frequency:

$$P(w_j) = \frac{N_j}{N}. \tag{3.59}$$

A decision can be made based only on the a priori probability by selecting the class w_j with the highest $P(w_j)$. However, this decision is obviously unreasonable because every string must be predicted to be one of the M classes of string lengths. A more precise decision is possible by considering further observations, such as the width of the string bounding box. Let x be a continuous random variable whose value is the width (in pixels) of the string bounding box, and $f(x \mid w)$ be a class-conditional probability function *pdf*. Given the *a priori* probability $P(w_j)$ and the *pdf* $f(x \mid w_j)$, we can compute the conditional probability $P(w_j \mid x)$ using the Bayes rule:

$$P(w_j \mid x) = \frac{f(x \mid w_j)P(w_j)}{f(x)} \tag{3.60}$$

where

$$f(x) = \sum_{j=1}^{M} f(x \mid w_j)\Pr(w_j). \tag{3.61}$$

Equation 3.61 is called the *a posteriori* probability as it is the probability of class w_j occurring after observing bounding box width x. Bayes rule shows how the observed data x changes the decision based on the *a priori* probability $P(w_j)$ using the *a posteriori* probability $P(w_j \mid x)$. Decision making based on the *a posteriori* probability is more reliable, because it employs both *a priori* knowledge and present observed data.

When large amounts of sample data are available, the probabilistic structure of the *pdf* can be directly estimated from the training data. However, available sample data are normally limited in practice. Thus, the *pdf* is usually assumed to have certain probabilistic structure. The Gaussian *pdf* is one of the most used *pdfs*. The univariate Gaussian *pdf* of the variable x for the class w_j is given by:

$$f(x \mid w_j) = \frac{1}{\sqrt{2p\, s_j^2}} \exp\left[-\frac{(x - m_j)^2}{2\, s_j^2} \right]. \tag{3.62}$$

where

$$m_j = \frac{1}{N_j} \sum_{k=1}^{N_j} x_{jk} \tag{3.63}$$

$$s_j^2 = \frac{1}{N_j} \sum_{k=1}^{N_j} (x_{jk} - m_j)^2 \tag{3.64}$$

and N_j is the number of observations which belongs to class w_j.

As described in [HUANG et al., 1990], there are different decision rules based on a posteriori probability. The most common are the Bayes and the minimum-error-rate rules.

3.3.1. Bayes decision rule

Let us consider the following loss function:

$$h(w_i \mid w_j) = \begin{cases} 0 & i = j \\ 1 & i \neq j \end{cases} \quad i, j = 1, ..., M \tag{3.65}$$

where M is the number of classes. This function is used to associate a unit loss to any error where $i \neq j$, while no loss is assigned to a right decision $i = j$. Using this scheme, we can compute the conditional risk associated with making decision w_i when the true class is w_j, as:

$$R(w_i \mid x) = \sum_{j=1}^{M} h(w_i \mid w_j) \Pr(w_j \mid x) \tag{3.66}$$

The objective of the Bayes rule is to minimize the overall risk involved in making a decision. For this purpose, we must to compute the risk in Equation 3.66 for each decision and select the one for which the conditional risk is minimum.

3.3.2. Minimum-error-rate decision rule

The risk corresponding to the loss function described in Equation 2.65 equals the average error probability. Another approach to make a decision is based on the minimum-error-rate decision rule, which can be used to minimize the average

probability of error. We can achieve the minimum error rate by selecting the decision that class w_i is correct, if the a posteriori probability $P(w_i \mid x)$ is a maximum.

3.4. Summary

In this chapter we have described the Hidden Markov Models, which have been successfully used to join segmentation and recognition in the field of speech recognition. This chapter has presented different types of HMMs taking into account the topology: ergodic and left-right models; and the observation probability distribution for each state: discrete, continuous and semi-continuous. The main algorithms used for training and scoring discrete first-order HMMs are presented. The Level Building Algorithm was described, since this dynamic programming technique is used to match single numeral models against an unknown numeral string in the implicit segmentation process of the SCB stage.

A quantization process based on the K-means algorithm was described. It is used to produce discrete observations from the feature space in both stages of the proposed method. It is necessary because the use of discrete HMMs. Finally, a brief description of Bayes theory was presented. In the proposed method, this theory is used to implement a classifier to predict the string length (number of digits) from the bounding box width (in columns) in order to deal with unknown length strings.

4. Preprocessing of Numeral Strings

In this chapter, we present the preprocessing techniques used to transfer a handwritten numeral string to a more appropriate form. The general objective is to reduce script variability, with specific objectives dependent on the proposed recognition method.

Section 4.1 focuses on the use of slant normalization in an implicit segmentation-based method for numeral string recognition. Different word slant normalization methods are presented and discussed. A word slant normalization method is selected and modified in order to improve the results for handwritten numeral strings. We assume that each connected component (CC) in the string has its own slant. The slant and contour length of each CC are used for obtaining the mean slant of the string. Both the original and modified methods are evaluated by means of some interesting analyses on the NIST SD19 database. These analyses show: a) the positive impact of slant correction on the number of overlapping numerals in strings; and b) the difference between normalizing isolated numerals based on the slant estimated from their own images and doing so based on the slant estimated from their original string images.

Section 4.2 describes a size normalization method which also takes into account an implicit segmentation-based strategy for recognizing numeral strings. This method is based on non-linear normalization in order to deal with inter- and intra-string size variations.

Finally, in Section 4.3, the method proposed in [SUEN et al., 1992] for smoothing binary images is described. This method is used to smooth the string contour in order to reduce the presence of spikes and notches caused by scanning noise or by the slant normalization method.

The analyses performed in this chapter also support the idea of using contextual information regarding string slant and intra-size variation to train the numeral HMMs of the SCB recognition stage.

4.1. Slant variation

The character inclination typically found in cursive script can be considered useless, notwithstanding the fact that it is writer's characteristic. Thus, it is common to encounter slant normalization in the preprocessing stage in writer-independent methods for word or string recognition. The general objective is to reduce script variability, with specific objectives dependent on the recognition approach. For example, in explicit segmentation-based methods, slant correction may be used to improve the results of segmentation techniques based on vertical single-run zones to define character candidates. In these methods, slant correction allows us to increase the detectability of a candidate to constitute a character. Another example is to be found in implicit segmentation-based methods, in which a vertical strip is used to scan the word image for feature extraction. Here, slant is corrected in order to minimize the problem of overlapping between adjacent characters, which interferes with the features computed from them.

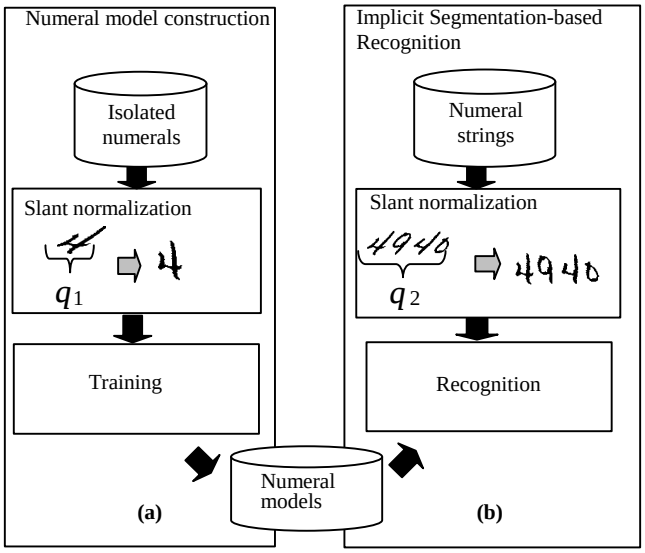


Figure 4-1 Slant normalization and a hypothetical implicit segmentation-based system

This work focuses on the use of slant normalization in an implicit segmentation-based method for numeral string recognition. Let us consider the system in Figure 4-1(b). In this example, the numeral strings are slant-normalized in order to reduce the overlap between adjacent numerals. To this end, a slope (q_2) is estimated from the whole numeral string image. String recognition is achieved by matching numeral HMMs against the normalized string by means of dynamic programming. The

construction of these models is presented in Figure 4-1(a). They are trained on isolated numerals, which are also slant-normalized. However, a different slope (q_1) is used, which is estimated from the numeral image. Although the same slant normalization method may be used to estimate q_1 and q_2, they may present a significant difference.

At this point, we can formulate the following questions:

1. What is the real impact of slant normalization on the number of overlapping numerals in the strings?
2. How different are q_1 and q_2?
3. If the difference between q_1 and q_2 is really significant, what is the real contribution to string recognition of considering some contextual information during slant normalization, *i.e.* correcting the slant of the training samples using the slope estimated from their original strings (q_2)?

To answer these questions, we first adapted a word slant normalization method to normalize numeral strings. Then, we performed some interesting analyses using naturally segmented numeral strings and corresponding digits extracted from the NIST SD19 database. These analyses were also used to compare the original and the modified slant normalization algorithms. The real contribution to string recognition of considering contextual information on the slant normalization of numerals used for training the HMMs, and also for dealing with intra-string size variation, are shown in the experiments reported in Chapter 6.

4.1.1. Slant normalization

The most important step in slant normalization concerns estimation. Since the characters in a word can present a non-uniform inclination, the methods for estimating slant are based on the mean inclination of these characters. Thus, only a mean correction of slant is possible. The objective of this correction is to transform the script image (a cursive word, for instance) by translating its points in the *x* dimension taking into account the estimated slope associated with the slant.

The slant correction methods described in this work were selected with a view to the possibility of using them on handwritten numeral strings. They represent different approaches to slant estimation; and the majority are strongly based on heuristics.

4.1.1.1. Brown & Ganapathy method

In [BROWN & GANAPATHY, 1983], slant estimation is obtained by placing two horizontal thresholds through the center of the word (see Figure 4-2). After the crossover points have been determined (points where the script crosses the thresholds), the associated slope is calculated from the crossover coordinates. The average of these slopes is used as the script slant. Slant correction is a linear transformation described by a 3x3 matrix, where only the *x* dimension is transformed.

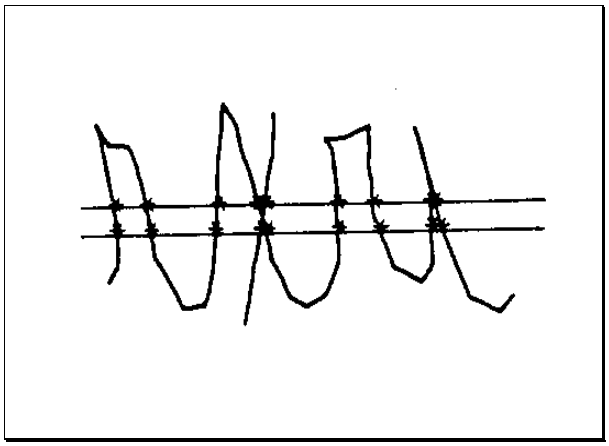


Figure 4-2 Cross pointers defined in two horizontal thresholds adapted from [BROWN & GANAPATHY, 1983]

4.1.1.2. Bozinovic & Srihari method

Bozinovic and Srihari proposed in [BOZINOVIC & SRIHARI, 1989] a method in which the first step consists of removing all horizontal lines which contain at least one run of length greater than a parameter *maxrun* (see Figure 4-3(b)). In addition, all horizontal strips lower than a parameter *stripheight* are removed (see Figure 4-3(c)). The second step consists of dividing the horizontal strips by vertical lines into isolated windows (see Figure 4-3(d)). For each window, the centers of gravity for its upper and lower halves are computed and connected (except windows with an empty half). Finally, the slope of the connecting line defines the slope of the window, and the average for all windows defines the slope b of the word.

Slant is corrected by applying the following transformation to each point with coordinates *x, y* in the original image:

$$x' = x - y \times \tan(b - def), \tag{4.1}$$

$$y' = y \tag{4.2}$$

where, *def* is used to specify the normal slant (90^o). An example of a slant-corrected word using the Bozinovic & Srihari method is shown in Figure 4-3(e).

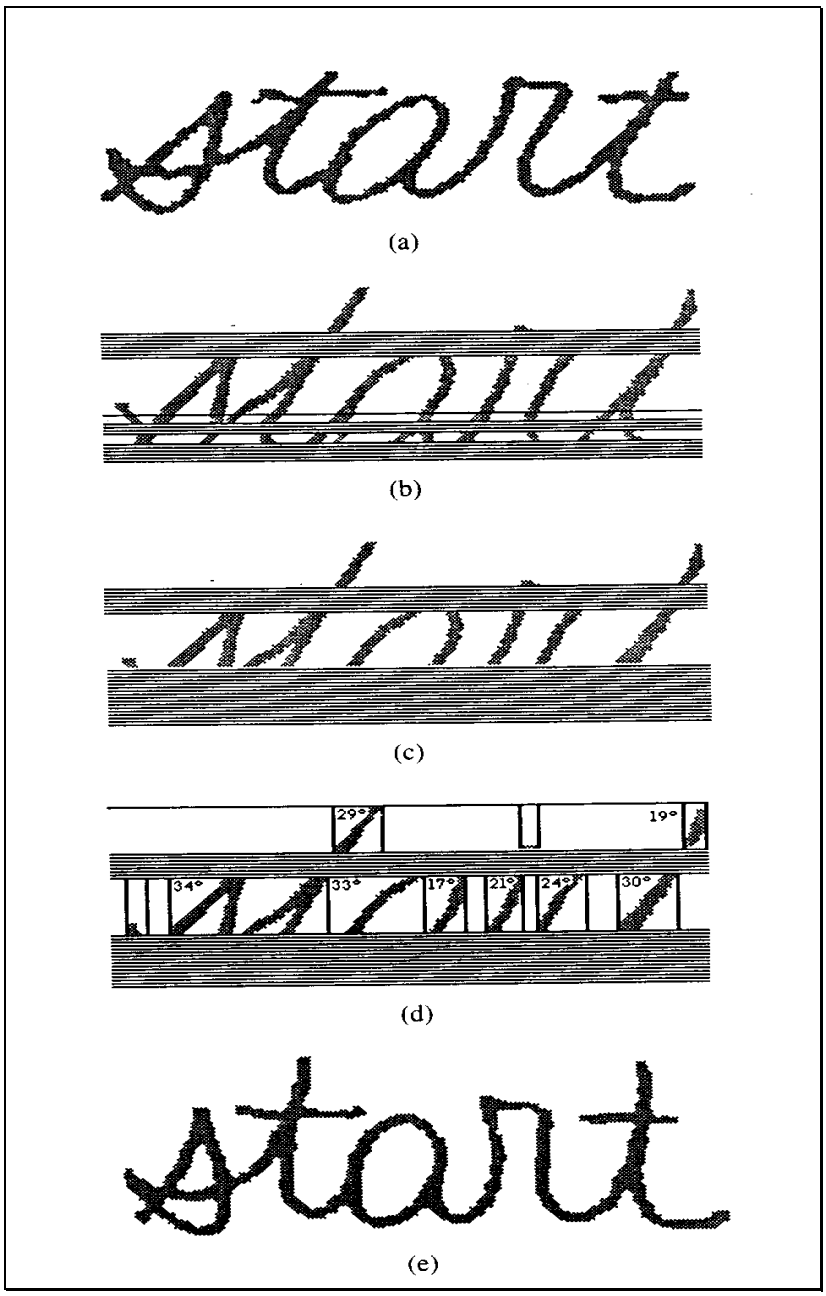


Figure 4-3 Bozinovic & Srihari Method a) word image; b) horizontal lines removed; c) small horizontal strips removed; d) strips divided by vertical lines into windows and slant angles of each window; e) slant-corrected image [BOZINOVIC & SRIHARI, 1989]

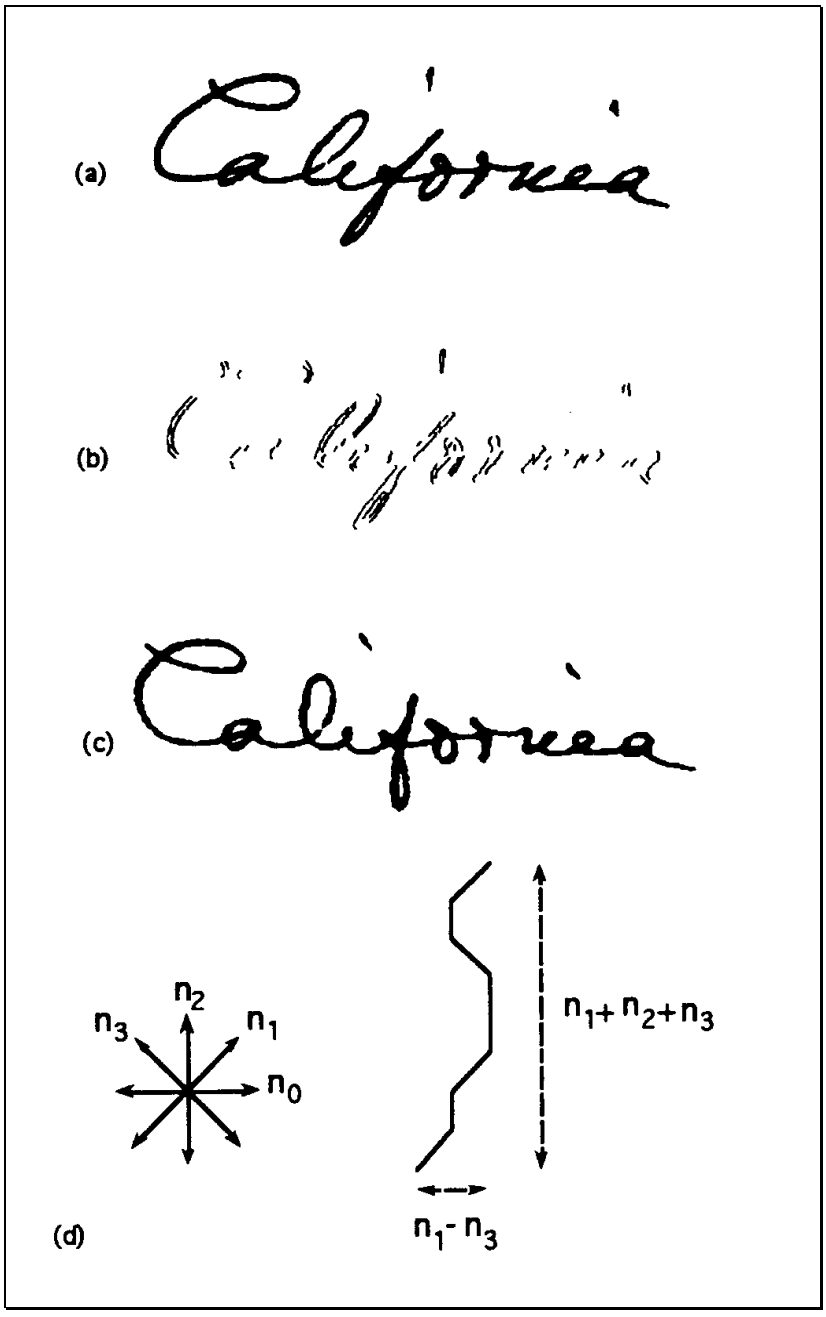


Figure 4-4 Kimura, Shridhar & Chen Method a) word image; b) chain code image without horizontal segments; c) slant-corrected image; d) a chain segment [KIMURA & SHRIDHAR, 1992]

4.1.1.3. Kimura, Shridhar & Chen (KSC) method

Kimura, Shridhar and Chen use the chain code of the word contour for slant estimation in [KIMURA et al., 1993]. Figures 4-4(a), 4.4(b) and 4.4(c) illustrate this method. The horizontal chain elements (code n_0) are not considered, and the other elements are divided into slant chain segments: n_1, n_2 and n_3, 45^o, 90^o and 135^o

respectively (see sample of a chain segment in Figure 4-4(d)). The average orientation of these segments is given by:

$$q = \tan^{-1}\left(\frac{n_1 + n_2 + n_3}{n_1 - n_3}\right) \tag{4.3}$$

4.1.1.4. Simoncini & Kovacs method

Simoncini and Kovacs method [SIMONCINI & KOVACS, 1995] for defining word slant is similar. Slant is estimated by means of the following expression:

$$q = \tan^{-1}\left(\frac{n_1 - n_3 + Kn_2}{n_1 + n_2 + n_3}\right) \tag{4.4}$$

where, again, n_1, n_2 and n_3 are the numbers of contour segments of 45^o, 90^o and 135^o respectively. $K \in [0,1]$ is an adjustment factor.

4.1.1.5. Yacoubi method

An interesting approach is proposed in [YACOUBI, 1996]. The author defines a number of lines with different inclinations (slope p_i) and the same origin, and which are regularly spaced from -45^o to 45^o, using multiples of an elementary angle ($= p/80$). Subsequently, a projection histogram H_i is calculated for each line. The H_i histogram is used to calculate a second histogram G_i, containing the number of segments in H_i for each possible length k. The surface S_i of each histogram G_i is calculated (or, the sum of the segments weighted by their respective lengths). Only segments of length greater than s are used (s = 3/5 x length of the word body). The mean slope is calculated as follows:

$$q = \sum_i c_i p_i \tag{4.5}$$

where:

$$c_i = \frac{S_i}{\sum_j S_j} \tag{4.6}$$

Figure 4-5 illustrates this slant correction method.

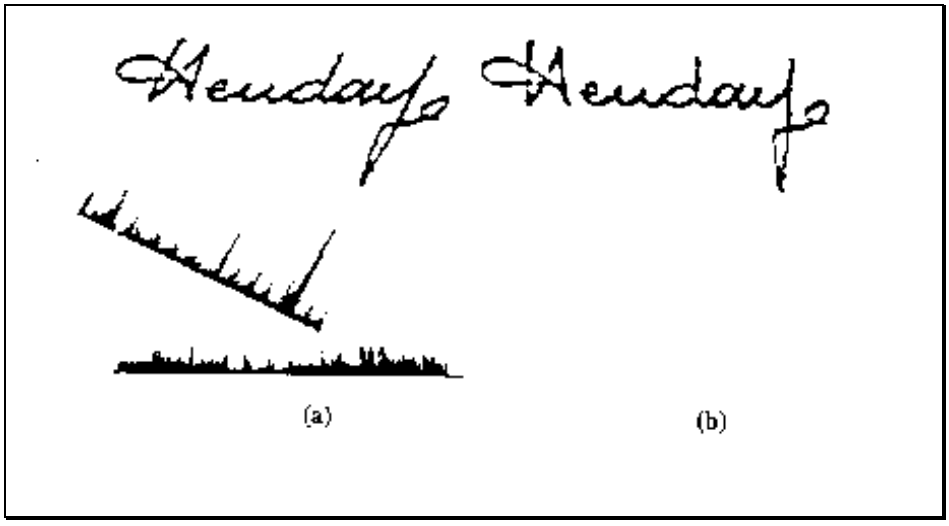


Figure 4-5 Yacoubi method; a) Projection histograms using inclination angle of 30^o and 0^o, respectively; b) word after slant correction.

4.1.2. Bozinovic & Srihari method vs Kimura, Shridhar & Chen method

The word image in [BROWN & GANAPATHY, 1983], [BOZINOVIC & SRIHARI, 1989] and [YACOUBI, 1996], as well as the chain code of the word contour in [KIMURA et al., 1993] and [SIMONCINI & KOVACS, 1995] have been used for slant estimation. Except for the KSC method, the majority of the methods described are strongly based on heuristics. Unfortunately, these empirical values have a direct influence on the estimation of the mean character inclination.

In order to evaluate slant correction methods for cursive words using numeral strings, two methods were chosen from among these methods. They were implemented and tested on handwritten numeral strings:

1. The Bozinovic & Srihari method, since it has been referenced many times in the literature. Recently this method was used in a numerical string recognition method in [NISHIWAKI & YAMADA, 1998].
2. The KSC method, since it does not use heuristics.

Slant-corrected handwritten numeral strings using both methods are shown in Figure 4-6. In [BOZINOVIC & SRIHARI, 1989], slant estimation is highly dependent on the parameter *maxrun*. During the tests and after several attempts, the *maxrun* and

stripheight parameters were fixed, at 15 and 8 respectively. However, these values were not suitable for all the string images (see Figures 4.6(b), 4.6(e), 4.6(f), and 4.6(j)). In fact, *maxrun* is dependent on the stroke width (thickness) of the string and must therefore be estimated for each string image in a particular way. An alternative is to define *maxrun* using a distance function to calculate the mean thickness of the string. However, this alternative probably represents problem in terms of time consumption.

There is only one disadvantage to the KSC method, which is that the image contour must be defined. However, no heuristic is used and the method yields good results (also in Figure 4-6).

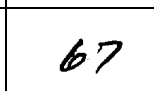
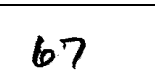
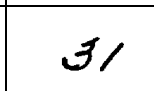
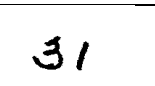
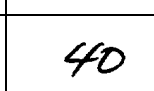
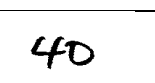
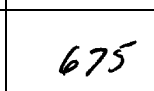
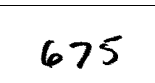
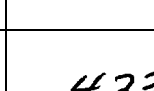
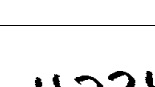
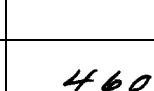
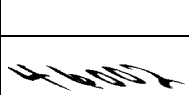
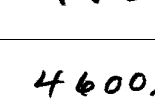
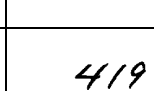
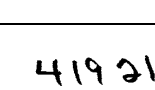
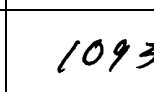
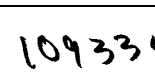
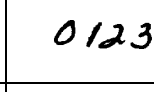
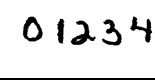
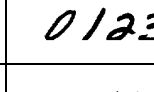
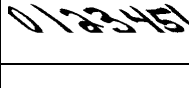
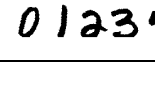
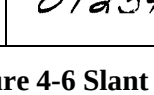
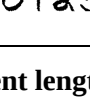
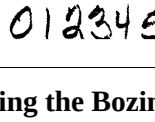
	Original Image	Bozinovic & Srihari method	Kimura, Shridhar & Chen method
A			
B			
C			
D			
E			
F			
G			
H			
I			
J			
K			

Figure 4-6 Slant correction of different length numeric strings using the Bozinovic & Srihari and Kimura, Shridhar & Chen methods

4.1.3. Proposal for a Modified Kimura, Shridhar & Chen (MKSC) method

This section describes an adaptation of the KSC method designed to improve its results for handwritten numeral strings. The cited method was originally proposed for the slant correction of cursive words. Thus, the slant is estimated from the whole word in a global way which is not satisfactory for numeric strings since these can be considered as a set of components (digits or fragments).

In fact, each string component can be considered a word with its own slant. In contrast, an independent correction of each component is not viable, as this may produce distortions in the numeral string because of the presence of broken digits.

One alternative is to estimate the mean slant based on the average of the slant of each string component. However, a simple arithmetic average can lead to serious problems. Again, the reason is the possibility of broken digits in the string. For example, the horizontal bar of a broken digit five can have a strong influence on the final string slant. This can be avoided by using a weighted average, where the slant estimation is based on the average of the slant of each string component weighted by its respective length (contour length).

Let *N* be the number of components (digits or parts of digits) in a numeric string; the mean slant can be estimated by the following expression:

$$\bar{q} = \frac{\sum_{i=1}^{N} (def - q_i) \times w_i}{\sum_{i=1}^{N} w_i} \tag{4.7}$$

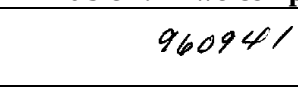
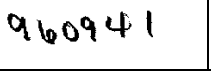
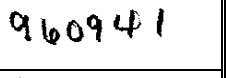
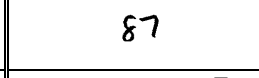
where *def* is the normal slant (90^o),

$$q_i = \tan^{-1}\left(\frac{n_{i1} + n_{i2} + n_{i3}}{n_{i1} - n_{i3}}\right) \tag{4.8}$$

and w_i is the chain code length of the i^{th} connected component.

Here, the slant of each string component is used to calculate the mean slant. The contribution of each component to the calculated average is weighted by its respective size (contour length). The objective is to avoid the distortions that can be generated by small fragments.

Table 4.1 Two comparisons between KSC and MKSC methods

			
Slant estimated for each component		Slant estimated for each component	
9	-45°	8	-26.6°
6	-45°	7	-45°
0	-26.6°	-	-
9	-26.6°	-	-
4	-45°	-	-
1	-45°	-	-
Slant results		Slant results	
KSC method	MKSC method	KSC method	MKSC method
Slant estimated: -45°	Slant estimated: -39°	Slant estimated: -26.6°	Slant estimated: -35.8°
			
Example 1		Example 2	

The examples in Table 4-1 show a comparison between the original KSC method and the modified method proposed. In both Example 1 and Example 2, the slant obtained from the adapted method (-39° and -35.8° respectively) is better than that obtained from the original method (-45° and -26.6° respectively), since these slants lie at a more uniform distance from the slant of each string component. Even visually, it is possible to verify a noticeable improvement. Figure 4-7 shows numeral strings of different lengths slant-corrected using both methods.

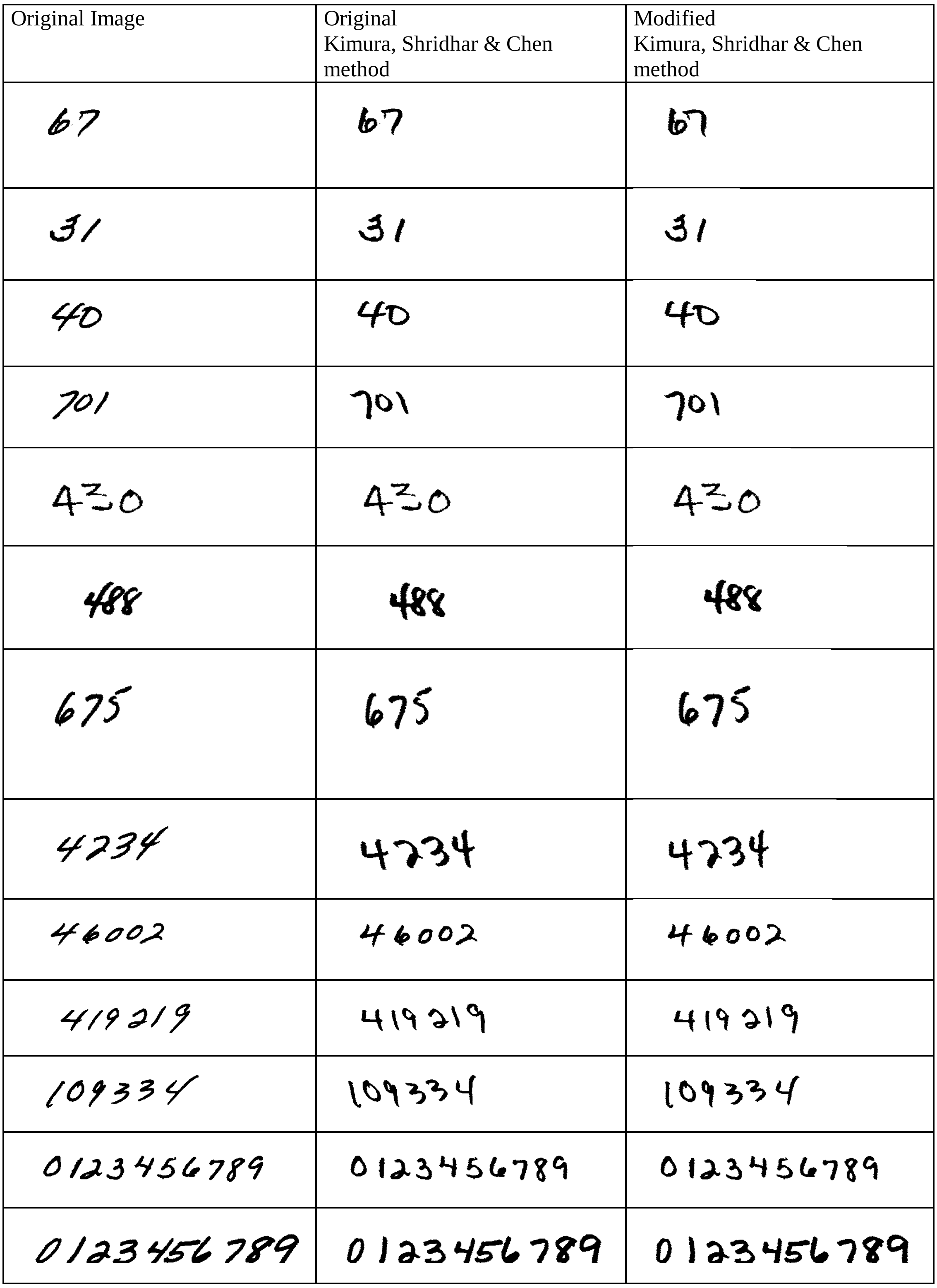


Figure 4-7 Numeric strings of different lengths slant-corrected using the original KSC and the MKSC methods

4.1.4. Preliminary analyses on the NIST SD19 database

Two interesting analyses using a data set extracted from NIST SD19 database are performed. In the first, we evaluate the difference between using and not using contextual information during slant correction of isolated numerals extracted from the strings. The second analysis consists of evaluating the impact of slant correction on the number of overlapping numerals in the strings.

An appendix in this work describes the process used to extract 44,256 naturally segmented strings, and 197,784 isolated numerals from the NIST database which are used in these analyses. A handwritten numeral string is considered "naturally segmented" when its components are recognized as isolated digits. Moreover, the recognition result of this string, from its digits, must correspond to that labeled by NIST. In this data set the original string of each isolated digit is known.

4.1.5. Impact of slant normalization on the number of overlapping numerals

One objective of this analysis is to estimate the impact of correcting string slant on the number of overlaps between adjacent numerals. Both the KSC and MKSC methods are used in this analysis.

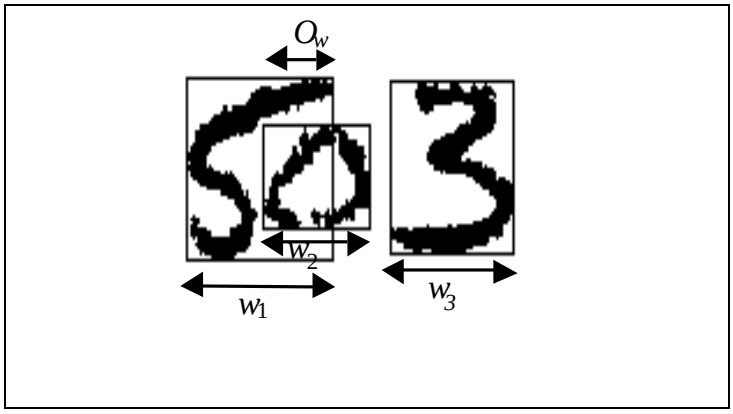


Figure 4-8 Overlap estimation

The overlap between adjacent numerals is estimated from the overlap between adjacent bounding boxes. For example, given the overlapping numerals in Figure 4-8, the algorithm calculates the overlapping percentage (Op_i) as:

$$Op_i = \frac{O_w}{w_i} \times 100 \tag{4.9}$$

In this example, the overlap (O_w) represents 47.6% of the width of the first bounding box (numeral 5), and 65.2% of the width of the second one (numeral 0). Figure 4-9 shows the impact of slant correction on the number of overlapping numerals in the 44,256 numeral strings analyzed. Before correcting the string slant, there were 9,425 numeral strings with overlapping numerals, which represent 21.29% of all the strings analyzed. After correcting the string slant using the KSC method, the number of strings with overlapping numerals was reduced to 4,767, which represents 10.77% of all strings analyzed.

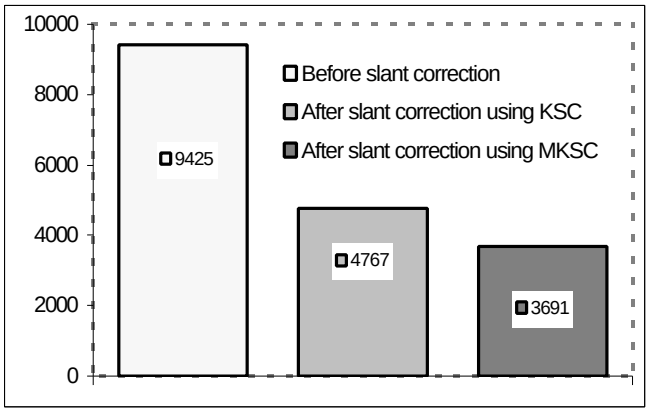


Figure 4-9. Number of overlapping numerals before and after slant correction using the KSC and MKSC methods

The number of overlapping numerals was reduced by 49.42%. On the other hand, using the MKSC method, the reduction is more significant. After string slant correction using this method, the number of strings with overlapping numerals was reduced to 3,691, which represents 8.34% of all the strings analyzed. The number of overlapping numerals was thus reduced by 60.83%.

4.1.6. Slant normalization with and without contextual information

The KSC and MKSC methods are also used to show the difference between two techniques for correcting the slant of isolated numerals extracted from strings: a) slant correction with contextual information, where the slant is estimated from the original string; and b) slant correction without contextual information, where the slant is estimated from each isolated numeral in particular, without taking into account its origin. The objective of this analysis is to evaluate the difference between these techniques in terms of the estimated slope. The isolated numerals extracted from the naturally segmented strings are used in this analysis (197,784 samples), since they have

a link to their original strings. For each numeral image, Δq is calculated, as $\Delta q = |q_1 - q_2|$, where q_1 is the slant of a numeral, estimated from its image; q_2 is the slant of a numeral, estimated from its original string image.

In addition, we have calculated the mean Δq and the dispersion for each numeral class. The results are shown in Table 4-2 for both slant correction methods. We can see that the algorithms for slant correction show similar results. However, the MKSC method provides a smaller dispersion than that obtained using the KSC method for all the numeral classes.

Table 4.2 ∆? mean and dispersion using the KSC and MKSC methods

Class	KSC method		MKSC method		Class	KC method		MKSC method	
	$m_{\Delta q}$	$J_{\Delta q}$	$m_{\Delta q}$	$J_{\Delta q}$		$m_{\Delta q}$	$J_{\Delta q}$	$m_{\Delta q}$	$J_{\Delta q}$
0	4.76	4.82	5.05	4.20	5	6.85	5.75	6.51	5.18
1	6.30	6.89	6.94	6.35	6	5.45	5.89	5.93	4.90
2	6.08	5.51	5.95	4.76	7	7.11	7.11	7.40	6.10
3	6.40	5.51	6.15	4.94	8	4.73	4.55	4.92	3.98
4	4.74	5.79	5.18	4.74	9	4.23	5.05	4.70	4.12
Global						5.66	5.83	5.88	5.07

Using the MKSC method, the mean variation becomes 5.88^o with a dispersion of 5.07^o, with the largest mean variation occurring in numeral class seven (7.40^o). Figure 4-10 shows that there is a significant number of cases where Δq is greater than 10.95^o ($m_{\Delta q} + J_{\Delta q}$). This number represents 11.74% of all the analyzed images (23,223 numeral images) using the MKSC method, and 15.23% of all the analyzed images (30,313 numeral images) using the KSC method. We can see that the modified method shows better results, since the slant estimated from the string is closer to that estimated from each isolated numeral. However, even using the MKSC method, the difference is still significant. In Table 4-3, we can see an example, where $\Delta q > (m_{\Delta q} + 2 \times J_{\Delta q})$ is based on the use of the MKSC method.

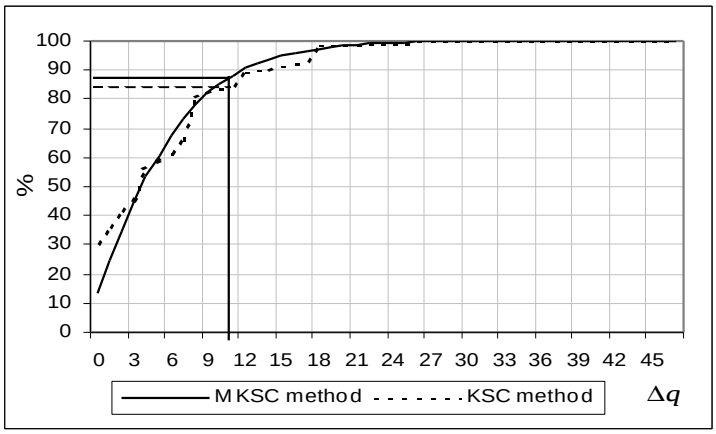
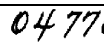
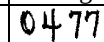


Figure 4-10 Cumulative distribution of Δq

Table 4.3 Example of a numeral string with significant Δq (17.2° and 21.6°)

Orig. Image	Slant-corrected Image(MKSC)	q_2	
		-35.6°	
Orig. Image	Slant-corrected Image	q_1	Δq
		-18.4°	**17.2°**
		-44.9°	9.3°
		-44.9°	9.3°
		-44.9°	9.3°
		-44.9°	9.3°
		-14.0°	**21.6°**

4.2. Size variation

Size variation may occur among different strings (*inter-string size variation*) or among the digits in the same string (*intra-string size variation*). Intra-string size variation (Intra-SSV), shown in Figure 4-11, generates blank spaces above and below some digits in the string. Intra-SSV is a real problem for methods based on implicit segmentation. Unlike explicit segmentation-based methods, no connected component detection is performed prior to the recognition process. This means that the feature extraction method must process the entire string image. So, we need to normalize the whole string. The size normalization method must deal with both inter- and intra-size variations.

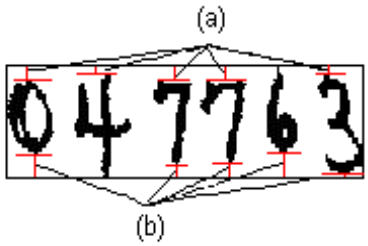


Figure 4-11 Intra-string size variation: (a) distance from the top of the bounding box; (b) distance from the base of the bounding box

A set of 197,784 isolated numerals is used to estimate height variation. The numeral height is obtained as a by-product of the process used to extract these isolated numerals from the naturally segmented strings in the training database (see Appendix). Figure 4-12 presents the height variation, ranging from 18 to 79 pixels. The mean height is 45 pixels. Figure 4-13 presents digit samples of different heights.

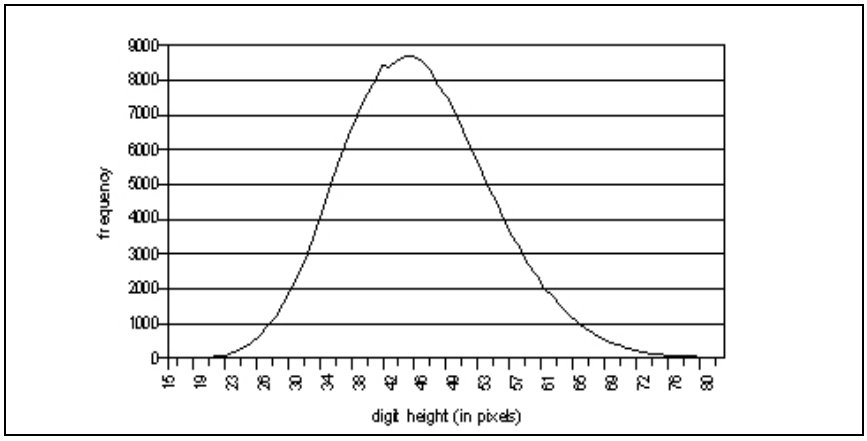


Figure 4-12 - Height variation

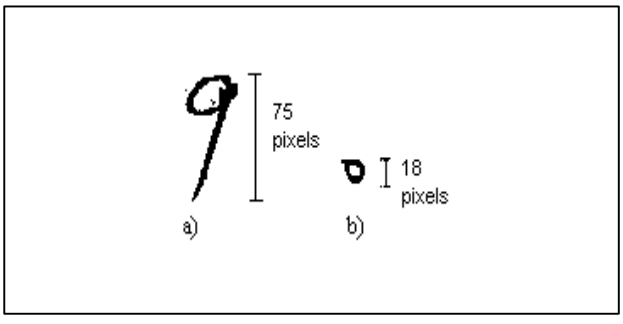


Figure 4-13 - Samples of height variation: a) height = 75 pixels; b) height = 18 pixels

After having observed this significant height variation on the numeral database, the numeral height variation inside each naturally segmented numeral string is

examined (44,256 samples). In order to perform this analysis, we first found the tallest numeral in the string. The height of this numeral (h_{max}) is considered as the numeral string height. Then, we calculated the difference between the height (h_i) of the other numerals in the same string and h_{max},, as:

$$\Delta h = \left| h_i - h_{\max} \right| . \tag{4.10}$$

The mean Δh ($m_{\Delta h}$) is 9.06 pixels with a dispersion ($J_{\Delta h}$) of 5.67 pixels.

Even considering just those Δh values greater than ($m_{\Delta h} + J_{\Delta h}$) as significant variations, a considerable number of numeral strings is observed, a total of 13,249 numeral strings corresponding to 29.93% of the analyzed strings. Some examples of numeral strings with $\Delta h >$ ($m_{\Delta h} + J_{\Delta h}$) are shown in Figure 4-14.

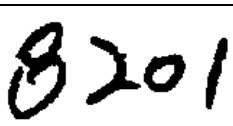
a)				
h_{max}= 67 pixels				
h_i	67	47	26	49
Δh	-	20	41	18
b)				
h_{max}= 65 pixels				
h_i	65		25	
Δh	-		40	
c)				
h_{max}= 62 pixels				
h_i	62		32	
Δh	-		30	

Figure 4-14 Examples of numeral strings with $\Delta h >$ ($m_{\Delta h} + J_{\Delta h}$)

These studies of numeral height variation confirm the necessity of defining size-invariant features or a non-linear size normalization method.

4.2.1. Linear size normalization

Linear size normalization only depends on the size of the input image and that of the normalized image. This linear transformation has the property of preserving linearity, and its calculation is easy [LEE & PARK, 1994].

If we let $I \times J$ denote the size of the input image and $M \times N$ the size of the normalized one, we can calculate the new position (m, n) of the normalized image as follows:

$$m = i \times \frac{M}{I} \tag{4.11}$$

$$n = j \times \frac{N}{J} \tag{4.12}$$

where, $i = 1,2,..., I$; $j = 1,2,..., J$ and $m = 1,2,..., M$; $n = 1,2,..., N$.

However, this kind of transformation is not sufficient to compensate for the size variation among the digits inside the string (Intra-SSV). In this case, a non-linear size normalization can be useful.

4.2.2. Non-linear size normalization

The method described is based on line density by crossing lines described in [LEE & PARK, 1994]. Let $F_H(i, j)$ and $F_v(i, j)$ denote two characteristic features of a pixel (i, j) in an image of size $I \times J$. If pixel (i, j) is a point in the background area, $F_H(i, j)$ and $F_v(i, j)$, are given by reciprocals of neighboring stroke distances, respectively, as shown in Figure 4-15.

$$F_H(i, j) = \frac{1}{h(i, j)} \tag{4.13}$$

$$F_v(i, j) = \frac{1}{v(i, j)} \tag{4.14}$$

where, $h(i, j)$ and $v(i, j)$ denote the horizontal and vertical distances between neighboring strokes respectively.

If pixel (i, j) is a point in the pattern area, $F_H(i, j)$ and $F_v(i, j)$ are given by very small values. Then, feature projection functions can be defined as follows:

$$H(i) = \sum_{j=1}^{J} F_H(i, j) \tag{4.15}$$

$$V(i) = \sum_{i=1}^{I} F_V(i, j) \tag{4.16}$$

The coordinates (m, n) of pixel (i, j) in the normalized image of size $M \times N$ are defined by:

$$m = A \sum_{k=1}^{i} H(k) \tag{4.17}$$

$$n = B \sum_{l=1}^{j} V(l) \tag{4.18}$$

where A and B are given by:

$$A = \frac{M}{\sum_{k=1}^{i} H(k)} \tag{4.19}$$

$$B = \frac{N}{\sum_{l=1}^{j} V(l)} \tag{4.20}$$

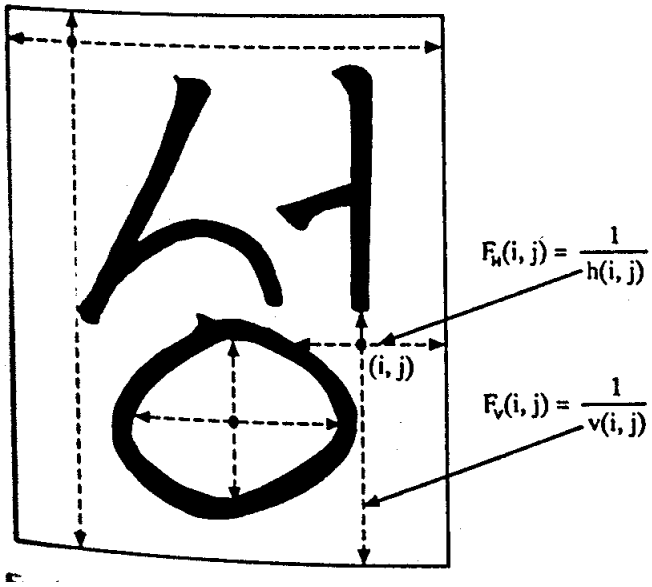


Figure 4-15 Characteristic features defined in [LEE & PARK, 1994]

4.2.3. Preliminary analysis of the size normalization method

In this preliminary analysis, the original string width and mean height of the analyzed digits (45 pixels) are used to define the size of the normalized strings. We did not change the string width in order that possible distortions on the digit strokes would

be reduced. Figure 4-16 shows the distribution of the intra-size variation (Intra-SSV in pixels) calculated as described in Section 4.2 for the 44,256 strings before and after applying the non-linear size normalization method.

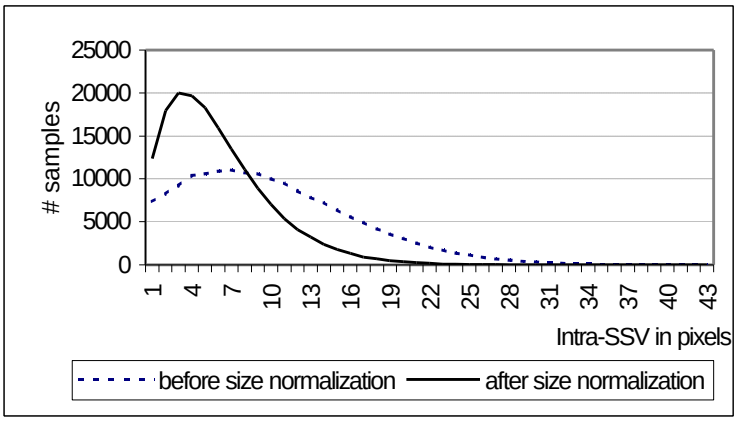


Figure 4-16 Intra-SSV before and after applying the non-linear size normalization method

Prior to the application of the size normalization method, a mean intra-size variation of 9.06 pixels and a deviation of 5.67 are observed. Following the application of this process, the mean drops to 6.08 pixels and the deviation to 4.46 pixels. Figure 4-17 shows the impact of the non-linear size normalization on mean intra-string size variation by string class.

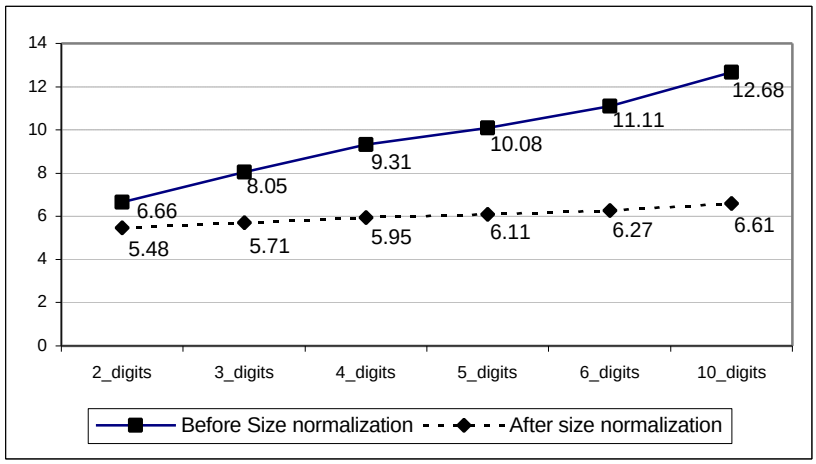


Figure 4-17 The impact of the non-linear size normalization on the mean intra-string size variation by string class.

4.3. Smoothing

The method proposed in [SUEN et al., 1992] is used to smooth the string contour by reducing the presence of spikes and notches caused by scanning noise. This method is based on a set of masks in a 3x3 window (see Figure 4-18). The central pixel

is filled when the window matches the patterns on mask (a), (b) or its equivalents with 90^o, 180^o and 270^o rotations. The central pixel is deleted when the window matches the pattern on the original masks, (c) or (d) or their equivalents, with 90^o, 180^o and 270^o rotations.

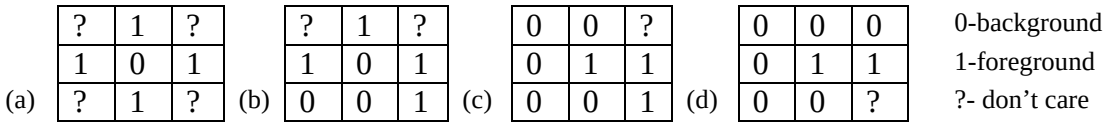


Figure 4-18 Masks used in the smoothing process

4.4. Summary

From the first analysis on the NIST database, it was possible to observe that slant normalization had brought about a significant reduction in the number of overlaps between adjacent numerals in strings. The MKSC method achieved a more significant reduction (60.83%) than that achieved by the KSC method (49.42%). This answers the first question in Section 4.1, in that it indicates that slant normalization is really helpful in reducing overlapping problems.

The second analysis has shown that there is a representative part of the analyzed data, referring to those cases where $\Delta q > (m_{\Delta q} + J_{\Delta q})$, in which there is a significant difference between the slants estimated with and without contextual information. In other words, the slant estimated from the isolated digit (q_1) and that estimated from its string (q_2) differ by more than 10.95^o in 11.74% of 197,784 analyzed digits using the MKSC method. This answers our second question, and at the same time justifies an investigation of the real contribution to string recognition of incorporating this contextual information in the slant normalization of the numeral training samples. Moreover, we observe that by using MKSC we reduce the number of cases where Δq is greater than $(m_{\Delta q} + J_{\Delta q})$. This means that we can approximate the slant estimated from the string to the slant estimated from its digits.

The answer to the last question related to the real contribution of these studies on the recognition performance of handwritten numerals is reported in the experiments in Chapter 6. These experiments show that the use of contextual information to provide the same conditions during training and testing is a promising strategy in implicit segmentation-based systems [BRITTO et al., 2000].

The preliminary analysis on size variation has shown that the non-linear size normalization method may reduce intra-string size variation. However, the experiments described in Chapter 6 show that contextual information related to Intra-SSV during training of the numeral models provides better recognition results.

5. Proposed Method

In this chapter, the proposed method for recognizing handwritten numeral strings, which can be categorized as a segmentation-free approach, is described in detail. Prior segmentation of strings into digits is avoided through the use of an implicit segmentation strategy and a further verification step. Section 5.1 gives a general overview of the method, which is composed of two HMM-based stages. Section 5.2 presents the first stage and its modules: Preprocessing, Foreground Feature Extraction (FFE) and Segmentation-Recognition (SR). The features and numeral HMMs are detailed. An evaluation of different HMM topologies in the LBA framework is presented. The numeral models are adapted in order to enable a string-based training by including an end-state in the HMM topology. These models show a better distribution of the observations among their states. This has brought about an improvement in terms of segmentation performance in the SCB stage. In this section as well, a space model built inside the numeral models is defined. This model is used for incorporating string contextual knowledge regarding inter-digit spaces into the numeral models of the SCB stage.

Finally, the Verification stage is detailed in Section 5.3. In this stage, a new set of features and numeral HMMs is used to improve the recognition performance of the entire system, and the best hypotheses generated by the SCB stage are re-ranked for this purpose.

5.1. Method overview

A general overview of the proposed method is presented in Figure 5-1. In the SCB stage, a given numeral string is first preprocessed in order to correct the slant, smooth the string contour and calculate the string bounding box. Subsequently, the FFE module scans the string image from left to right, while a feature vector based on foreground information is calculated for each column in the string bounding box. This vector is mapped to a discrete symbol available in a previously constructed codebook. The output of the FFE module is a sequence of discrete observations representing the entire

numeral string. The length of this sequence corresponds to the number of columns in the string bounding box.

In the SR module, numeral models trained on isolated digits (l_c^0, l_c^1,..., l_c^9), but considering string contextual information, are matched to the observation sequence provided by the FFE module. The objective is to find the *N* best segmentation-recognition paths or hypotheses using an implicit segmentation-based strategy. For each segmentation-recognition hypothesis, the following information is provided: string length (number of digits), segmentation points, recognition result of each string segment and the global string recognition probability.

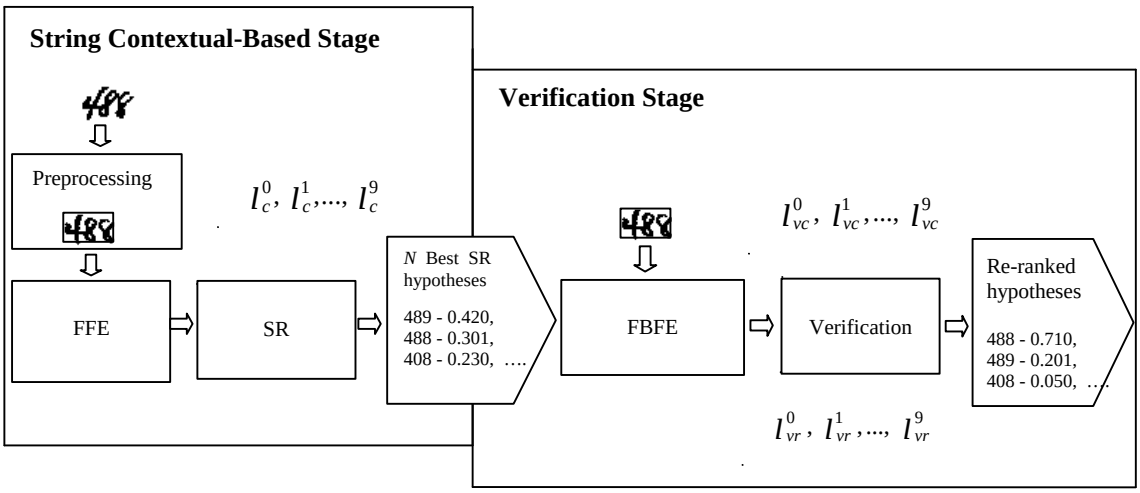


Figure 5-1 General overview of the proposed method

The segmentation-recognition hypotheses generated by the SCB stage are re-ranked in the Verification Stage. Basically, this second stage consists of an HMM-based digit classifier trained on isolated digits without taking into account any string contextual information. A new set of features combines foreground and background information to improve the recognition performance of the numeral HMMs. Moreover, 10 additional numeral HMMs (l_{vr}^0, l_{vr}^1,..., l_{vr}^9) based on the rows of the numeral images are combined with the column-based models (l_{vc}^0, l_{vc}^1,..., l_{vc}^9) to ensure an accurate representation of the digit classes.

The verification process starts in the Foregroung/Background Feature Extraction (FBFE) module, in which the segmentation points provided by the first stage are used to define string segments and calculate their bounding boxes. Then, for a given segment, a feature vector combining foreground and background information is extracted for each

column of the segment bounding box. This vector is mapped to a discrete symbol in a previously constructed codebook. A similar process is carried out for the rows of the segment. The output of the FBFE module consists of two sequences of discrete observations for each segment: column-based and row-based sequences. In the Verification module, the first step is to select the column (l_{vc}^{0}, l_{vc}^{1},..., l_{vc}^{9}) and row (l_{vr}^{0}, l_{vr}^{1},..., l_{vr}^{9}) models corresponding to the segment to be verified. This is done by using the recognition result of each string segment provided by the SCB stage. The selected models are scored using the corresponding sequence of observations extracted by the FBFE module. The output probabilities of both column and row models are combined by summing their logs. The resulting probability corresponds to the probability of the string segment being correct. A string probability is obtained by summing the probability of each segment of the numeral string. This string probability is added to the probability of the corresponding segmentation-recognition hypothesis obtained from the SCB stage. The final resulting probability is used to re-rank the string segmentation-recognition hypothesis.

5.2. String Context-Based (SCB) stage

As previously stated, the general objective of this stage is to provide the *N* best segmentation-recognition paths or hypotheses for a given numeral string. To this end, it is composed of three modules: Preprocessing, Foreground Feature Extraction (FFE) and Segmentation-Recognition (SR). The main characteristic of this stage is the use of an implicit segmentation based strategy to integrate the segmentation and recognition processes. It has been shown that this is a promising strategy for dealing with the string difficulties presented in Chapter 1, since this allows us to avoid a prior segmentation of the string into digits without the aid of a recognizer. In addition, this stage contains numeral HMMs trained on isolated digits, but considering contextual information (CI) regarding slant and intra-string size variations. The main purpose of using such a CI is to ensure an accurate representation of numeral strings.

5.2.1. Preprocessing module

In this module, string slant is corrected in order to reduce script variability. The method proposed in Chapter 4 has also been shown to be really helpful in alleviating

overlapping between adjacent digits which may interfere with the column-based features extracted from them. The smoothing method described in [SUEN et al., 1992] is used, before and after slant correction, to reduce possible artifacts on the string contour. The last step of this module concerns the calculation of the string bounding box.

A size normalization method was also evaluated in Chapter 6, and the non-linear size normalization method described in Chapter 4 produced a significant reduction on the intra-string size variation. However, the experimental recognition results have shown that training the numeral models considering contextual information is more appropriate for dealing with this string difficulty, since it avoids stroke distortions or new touching cases caused by using a size normalization method.

5.2.2. Foreground Feature Extraction (FFE) module

In the implicit segmentation strategy of the SCB stage, a preprocessed string image is scanned from left to right, while numeral HMMs are matched to it by means of the LBA described in Chapter 3. Thus, it is necessary to compute a feature vector as a function of an independent variable through the use of a windowing scheme.

These schemes are common in speech recognition, where the speech signal is divided into a sequence of frames, and a feature vector is computed for each frame [RABINER, 1989]. A narrow vertical strip has been used in a similar way in off-line recognition systems. The image of a character, word or text to be recognized is scanned, and, for each horizontal position, a feature vector is computed. Different schemes have been proposed in the literature.

In [MAKHOUL et al., 1998], the authors propose a script-independent methodology for optical character recognition. In this work, after skew and rotation corrections, and character height normalization, a line of text is scanned by a narrow vertical strip from left to right or right to left. The latter scan sense is used for Arabic script. During the scan process, a feature vector is computed for each horizontal position. The window, called a frame, is defined with a width of 1/15 of the text line height. The overlap from one frame to the next is defined as 2/3 of the frame width. Each frame is still divided into 20 vertically overlapping cells, and the following features are computed:

- intensity, *i.e.* the percentage of black pixels within each cell;
- vertical derivative of intensity across vertical cells;
- horizontal derivative of intensity across overlapping frames;
- local slope and correlation across a window two cells square.

Figure 5-2 shows a line of Arabic text and the frame divided into cells as used in [MAKHOUL et al., 1998] for feature extraction.

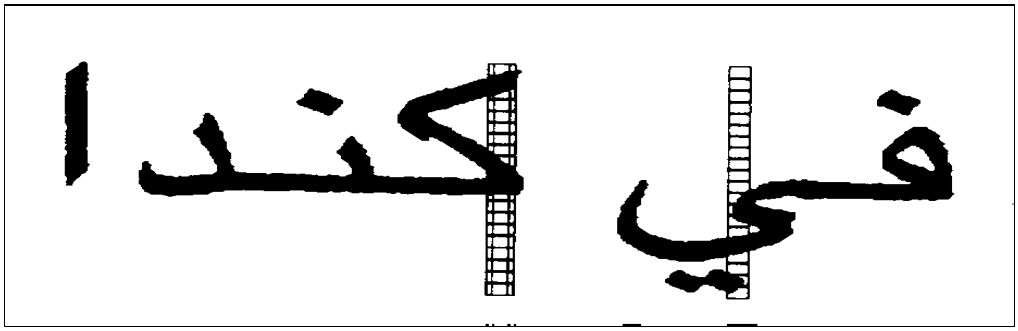


Figure 5-2 Frame divided into cells, as proposed in [MAKHOUL et al., 1998]

In [GUILLEVIC & SUEN, 1997], the authors scan the word image using a sliding window, and, for each position, a feature vector is extracted. The width of the window is defined as a fixed fraction of the main body height (distance between upper and lower base lines), and the overlap from one window to the next is fixed at 50% of the width. Each window is subdivided into 5 horizontal regions, corresponding to the ascender part of the word (j_0), the upper main body (j_1), the medium area (j_2), the lower part of the main area (j_3), and the descender area (j_4). Figure 5-3 shows the five regions of interest defined.

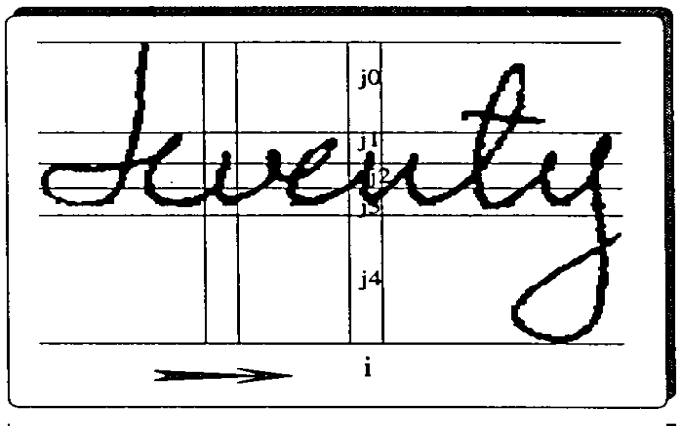


Figure 5-3 Frame divided into five regions, as proposed in [GUILLEVIC & SUEN, 1997]

The contour information of the word is used to extract the number of contour points in the four directions given by the Freeman code: 0, 45, 90 and 135 degrees. These 4 slope features are computed for each of the 5 regions, for a total of 20 features. In addition to these slope features, a position feature is computed for the ascender and descender regions as the average position of all contour points.

In another scheme, described in [ELMS, 1996], a vertical shape profile is proposed in order to recognize polyfont printed characters, and in which features are extracted from columns of pixels. Here, the width of the window is one pixel (a column). A shift-invariant feature, called the shape feature, represents a pattern of bits in the window, and a COG feature represents the relative center of the pixel mass with respect to the previous windows. These shape features are quantized using a codebook, which contains 32 possible bit patterns. A shift-invariant Hamming distance measure is defined to perform the shape quantization.

Similar to [ELMS, 1996] in our windowing scheme is a feature vector calculated for each column of the digit or string image. With this scheme, the problem of defining the size of a frame or sliding window is avoided, as is the overlap from one frame to the next, such as in [MAKHOUL et al., 1998] and [GUILLEVIC & SUEN, 1997]. We took this decision considering the preliminary analyses on the NIST database presented in Chapter 4, where it is possible to observe significant inter- and intra-string size variations. In this case, to define a frame and divide it into cells, it is necessary to consider a non-linear size normalization method to deal with both inter- and intra-size variations. However, such a scheme may cause distortions in the digit strokes or broken digits, or may even generate new touching cases. To compensate for not using such a zoning scheme, the relative position of each feature from the top of the digit or string bounding box is computed. The relative position in this case is size-invariant and enables simulation of a zoning scheme.

The FFE module computes a feature vector composed of 34 foreground features. This 34-feature vector is mapped to one of the 256 possible discrete symbols available in the codebook previously constructed using the vector quantization process described in Chapter 3. The output of the FFE module is a sequence of discrete symbols representing a given numeral string. The length of the sequence corresponds to the width (number of columns) of the string bounding box.

Foreground features

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

The proposed foreground feature set is composed of local and global features extracted from each string column. The local features are based on transitions from background to foreground pixels, and *vice versa*. For each transition, the mean direction and corresponding variance are obtained by means of the statistical estimators defined in [SABOURIN, 1990]. These estimators are more suitable for directional observations, since they are based on a circular scale. For instance, given the directional observations $a_1 = 1^o$ and $a_2 = 359^o$, they provide a mean direction ($\bar{a}$) of 0^o instead of the 180^o calculated by conventional estimators.

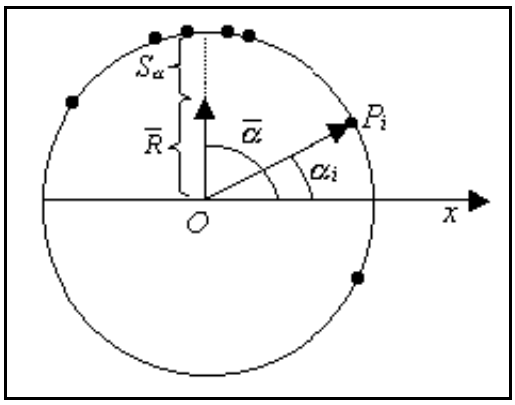


Figure 5-4 Circular mean direction $\bar{a}$ and variance S_a for a distribution $F(a_i)$

Let $a_1,...,a_i,...,a_N$ be a set of directional observations with distribution $F(a_i)$ and size *N*. Figure 5-4 shows that a_i represents the angle between the unit vector $\overline{OP_i}$ and the horizontal axis, while P_i is the intersection point between $\overline{OP_i}$ and the unit circle. The cartesian coordinates of P_i are defined as:

$$(\cos(a_i), \sin(a_i)) \qquad (5.1)$$

The circular mean direction $\bar{a}$ of the *N* directional observations on the unit circle corresponds to the direction of the resulting vector $(\bar{R})$ obtained by the sum of the unit vectors $(\overline{OP}_1, ..., \overline{OP}_i, ..., \overline{OP}_N)$. The center of gravity $(\bar{C}, \bar{S})$ of the *N* coordinates $(\cos(a_i), \sin(a_i))$ is defined as:

$$\bar{C} = \frac{1}{N}\sum_{i=1}^{N}\cos(a_i) \tag{5.2}$$

$$\bar{S} = \frac{1}{N}\sum_{i=1}^{N}\sin(a_i) \tag{5.3}$$

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

$$\bar{R} = \sqrt{(\bar{C}^2 + \bar{S}^2)} \tag{5.4}$$

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

$$\cos(\bar{a}) = \frac{\bar{C}}{\bar{R}} \tag{5.5}$$

$$\sin(\bar{a}) = \frac{\bar{S}}{\bar{R}} \tag{5.6}$$

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

$$S_a = 1 - \bar{R} \qquad 0 \le S_a \le 1 \tag{5.7}$$

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

In addition to this directional information, we have calculated two other local features: a) relative position of each transition with respect to the top of the digit

bounding box; and b) whether the transition belongs to the outer or inner contour, which shows the presence of loops in the numeral image. The first feature is used to compensate for the lack of a zoning scheme in our feature extraction method. The second is used to detect the presence of loops, which are very discriminative for handwritten numerals. After checking several digit images, we defined a maximum of 8 transitions per column. Thus, the feature vector is composed of 32 features at this point.

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

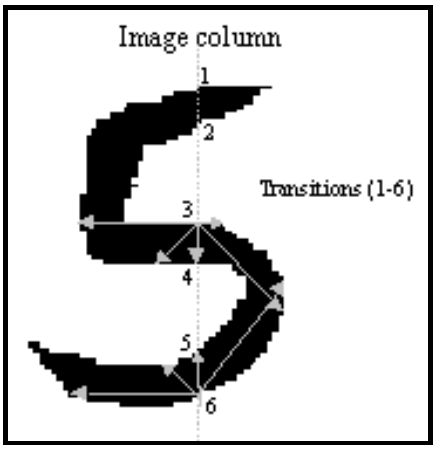


Figure 5-5 Transitions in a column image of numeral 5, and the directional observations used to estimate the mean direction for transitions 3 and 6

5.2.3. Segmentation-Recognition (SR) module

This module integrates segmentation and recognition through the use of an implicit segmentation-based strategy. It does so using the LBA described in Chapter 3, which is responsible for matching numeral HMMs to the preprocessed string represented as a sequence of discrete symbols provided by the FFE module.

To ensure an accurate representation of a numeral string, the numeral HMMs (l_c^0, l_c^1,..., l_c^9) are trained on a special data set extracted from the original NIST SD19. To create this data set, an automatic process based on the digit classifier is described in the appendix of this work and also in [BRITTO et al., 1999]. In this process, a

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

Contextual information used during training concerns string slant and intra-string size variations. The contribution to string recognition of the use of the slant estimated from the original string to correct the isolated digits used for training an implicit segmentation-based system is presented in Chapter 6 and also in [BRITTO et al., 2000].

The intra-string size variation generates blank spaces on the top and on the bottom of some digits in the string, as shown in Figure 5-6.

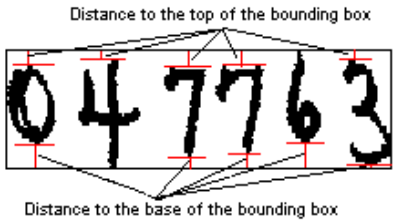


Figure 5-6 Intra-string size variation

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

The structure of the numeral HMMs in our baseline SCB stage was experimentally defined. Some preliminary results were obtained using a left-right discrete HMM with 5 states. This structure was optimized taking into account the methodology defined in [WANG, 1994] and described in Chapter 3. The K-means algorithm, also described in Chapter 3, is used to evaluate different codebook lengths. The best results were achieved using a codebook with 256 entries. All experimental results are shown in Chapter 6.

5.2.4. Numeral models

The topology of the numeral models is defined taking into account the recognition of handwritten text and the use of LBA. This means a left-right model without initial or end-states. The number of states was experimentally defined. Figure 5-7 shows the initial 5-state HMM topology used in the baseline SCB stage.

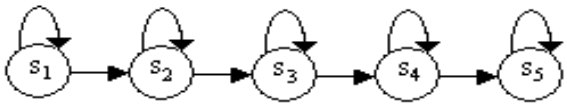


Figure 5-7 Left-to-right HMM model with 5 states

The same, or similar, topology can be found in related works. In [ELMS et al., 1998], the authors use it to model character classes to recognize fax-printed words. The same structure is used for modeling numeral classes to recognize handwritten numeral strings in [PROCTER & ELMS, 1998]. Another similar HMM topology, with additional skip transitions, is used for modeling airlines vocabulary in [RABINER & JUANG, 1993]. In all the above, the LBA is used as a recognition algorithm.

As we can see, the HMM topology used in the baseline system does not present additional states (initial or end-states) to enable the concatenation of numeral models, since these are not necessary in the LBA framework. In the SR module, 10 numeral models independently trained on isolated numerals are used to recognize strings, and the LBA is responsible for finding the best sequence of these models for a given numeral string. However, this kind of topology does not allow us to model the interaction between adjacent numerals in strings. Moreover, in the preliminary experiments on numeral strings in Chapter 6, we can observe a significant loss in terms of recognition performance as the string length increases (see Figure 6-1). In order to better understand the behavior of these numeral models, the distribution of observations among the HMM states is computed during their training on 50,000 isolated numerals (5,000 samples per class). Figure 5-8 presents these distributions as a *5-state HMM without end-state*.

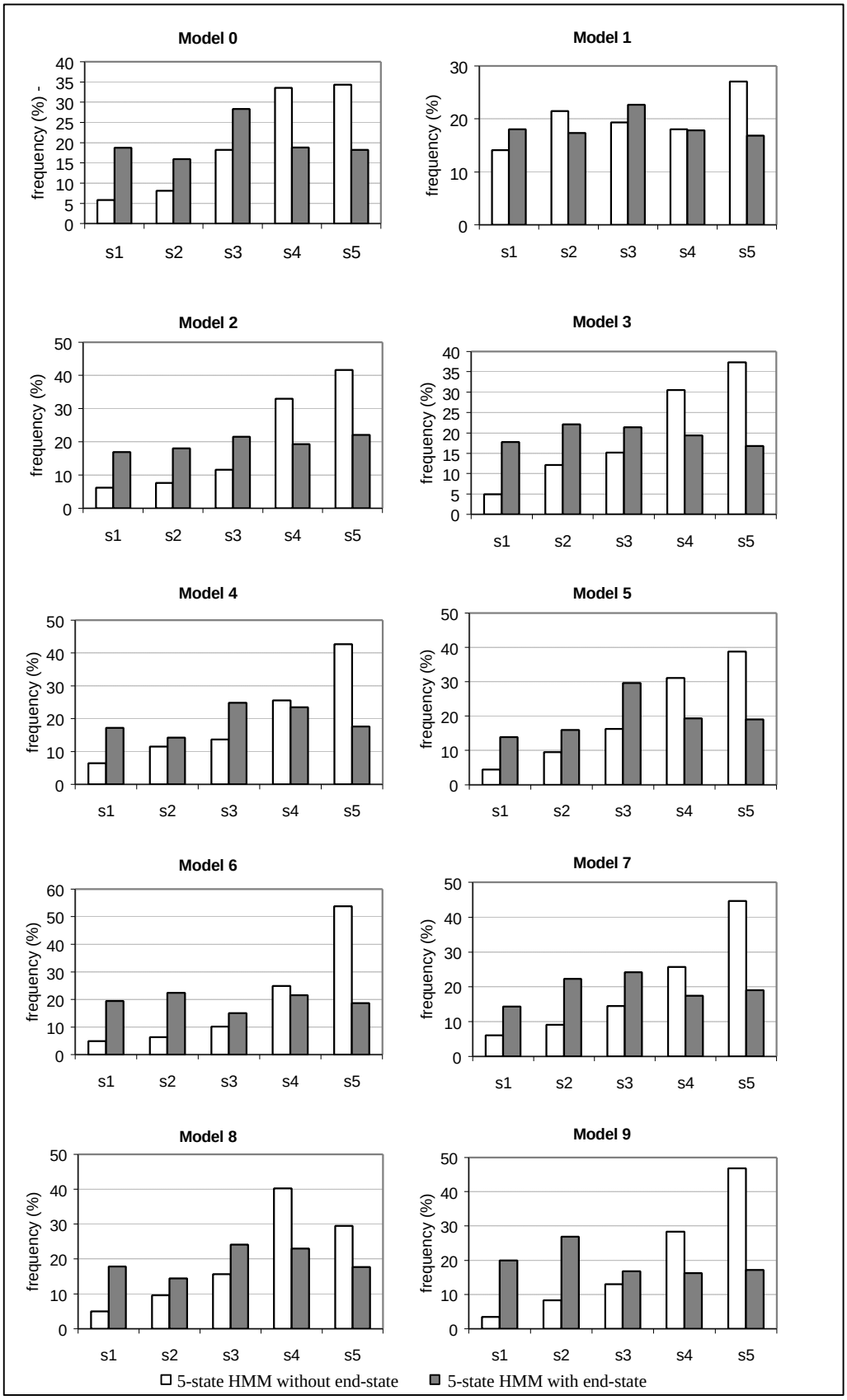


Figure 5-8 Distributions of observations among the HMM states computed during model training

These unbalanced distributions of observations among the states, associated with the presence of a self-transition with probability value equal to 1.0 in the last HMM state (s_5), have a negative impact on the segmentation performance of the SR module. To better explain, let us consider paths A and B in the LBA trellis in Figure 5-9, which share the same pathway until time t=4. Path A reaches state 5 (s_5) first (at time t=6). Meanwhile, path B may not reach the final state, even though it is a promising path. This may occur because the transition probability from state 4 to state 5 (a_{45}) (a small value because of the nature of the distributions observed) must compete with the self-transition probability on state 5 (equal to 1.0, since there is no transition out of this state). Under this condition, the numeral recognition at this level may succeed, although without representing the best segmentation path. This non-optimum matching can bring about problems for the next LBA levels. This explains why, in the baseline system, the recognition of numeral strings drops drastically as their length increases (see Figure 6-1 in Chapter 6).

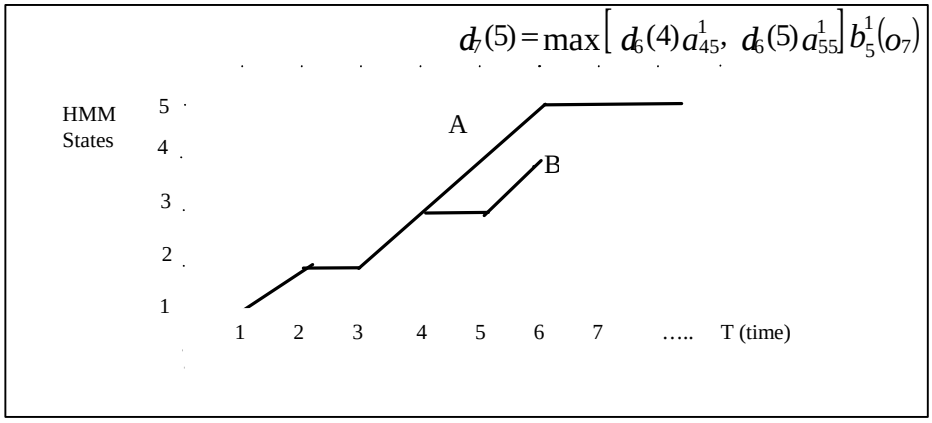


Figure 5-9 Paths A and B in an LBA level considering model l_1

To deal with this problem and also adapt the numeral models to a string-based training, we include an end-state in the HMM topology (see Figure 5-10). The new models show a better distribution of the observations among their states, as we can see in Figure 5-8 *(5-state HMM with end-state*), and avoid a self-transition with probability value equal to 1.0 in the state 5 (s_5). The end-state does not absorb any observation, and it is useful to concatenate the numeral models during a string-based training. The positive impact of this modification on the HMM topology to the string recognition is shown in Section 6.2.4 of the Chapter 6 and also in [BRITTO et al., 2001a].

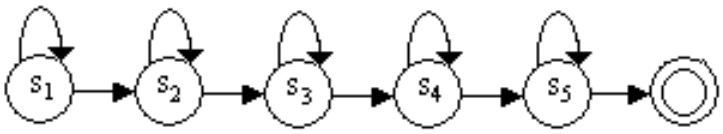


Figure 5-10 5-state HMM with an end-state

Based on this new topology, we can pay some attention to the possibility of integrating handwriting-specific knowledge into the model structure to obtain an accurate representation of numeral strings.

In addition, the final length of each numeral HMM (number of states) was redefined through the use of the scheme described in Chapter 3. This scheme estimates a range for the length of each numeral HMM. The final length of each model in Table 5-1 is experimentally defined (see Chapter 6).

Table 5.1 Final length of each numeral HMM

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

5.2.5. Space model

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

The strategy of building the space model into the numeral models instead of using an independent model is an important one in the LBA framework, since possible future problems when the method is generalized for unknown-length strings will be

avoided. An independent space model would represent one more model competing at each LBA level, which must be taken into account to estimate the string length (number of digits).

The space model is trained on digit pairs extracted from the NIST database. In this training, for a given digit pair the corresponding numeral models are concatenated by using the end-state. In fact, the end-state of the first model is replaced with the first state of the second model (see Figure 5-11).

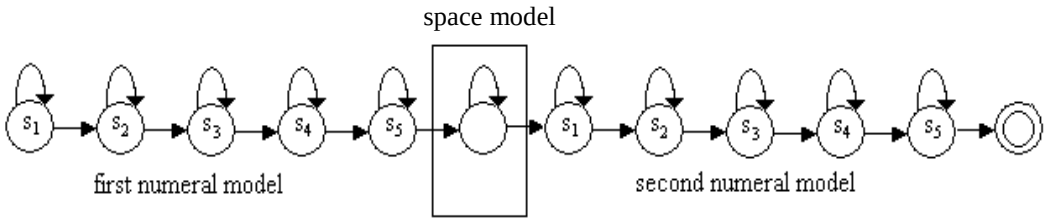


Figure 5-11 Concatenation of numeral models during string-based training

For the space model experiment, we use a two-step training mechanism: 1) 10 numeral models are trained first on isolated digits; 2) the numeral models are submitted to a string-based training using digit pairs (DPs) extracted from the NIST database. The DP database is balanced in terms of number of naturally segmented, overlapping and touching numerals. The NIST series hsf_0 to hsf_3 were used to provide the training and validation samples. Table 5-2 shows the number of samples in the training and validation sets, and also presents the number of samples representing each string class.

Table 5.2 Digit pair database (strings composed of 2 digits)

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

We use the two-step training mechanism described above to evaluate the following strategies: 1) the use of one space model for each numeral class; and 2) the use of one space model representing all numeral classes. In both, just the space model parameters are estimated during the second step of the training. The parameters corresponding to the numeral models are kept the same as estimated during the first

training step based on isolated numerals. The corresponding experimental results are reported in Chapter 6.

5.3. Verification stage

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

5.3.1. Foreground/Background Feature Extraction (FBFE) module

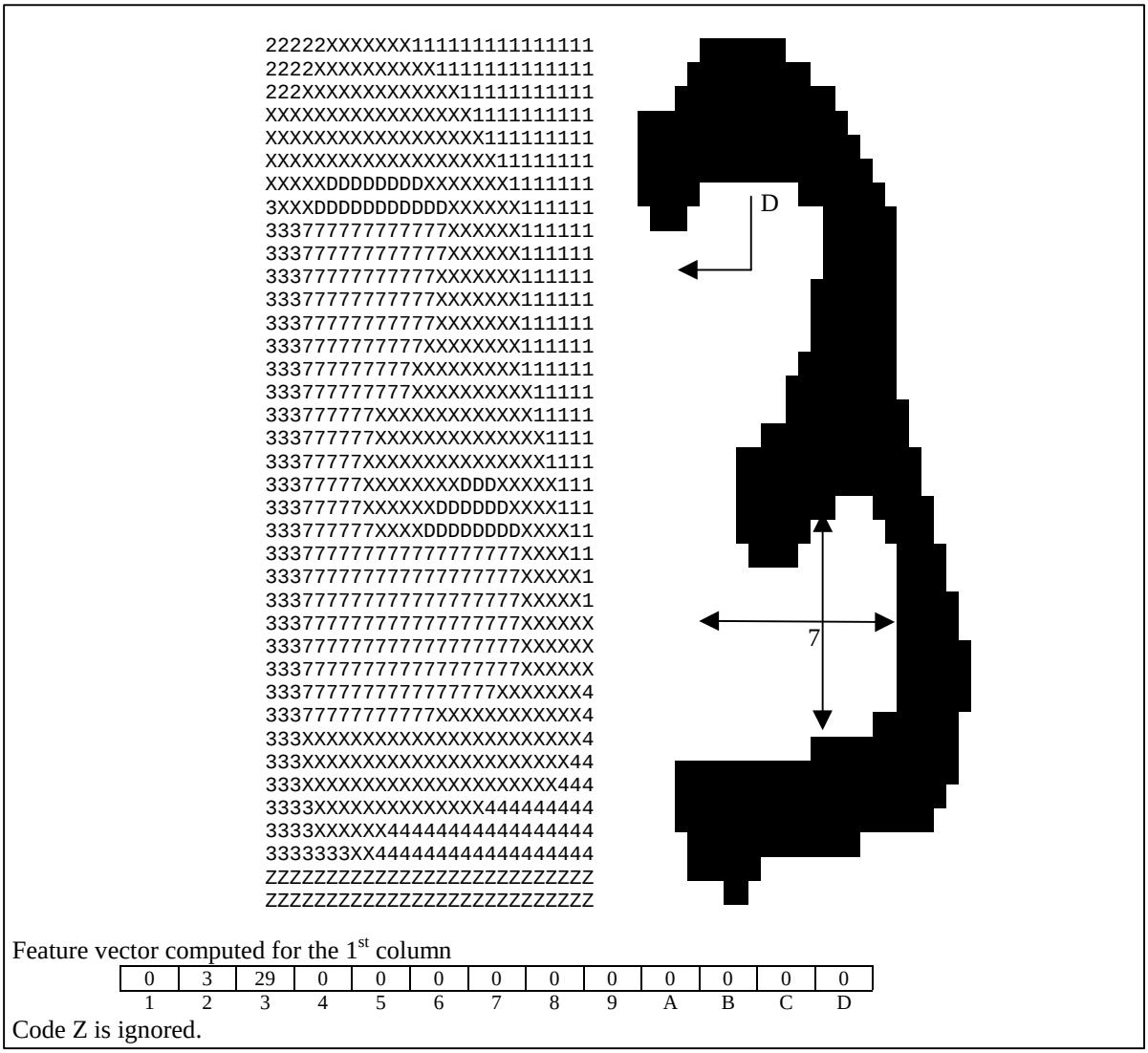
The verification stage starts in this module, in which the segmentation points provided by the first stage are used to define string segments and calculate the corresponding bounding boxes. Then, for a given segment, this module extracts a feature vector combining foreground and background information for each column in the segment bounding box. This feature vector is mapped to one of 256 possible discrete symbols available in a codebook previously constructed using the method described in Chapter 3. A similar process is carried out for the rows of the segment. Thus, the output of the FBFE module consists of two sequences of discrete observations for each segment: column-based and row-based sequences.

The feature vector computed for each column and for each row of the digit image is composed of 47 features: the 34 foreground features used in the SR module, plus 13 background features.

Background features

The background features are based on concavity information. These features are used to highlight the topological and geometrical properties of the digit classes. Each

concavity feature represents the number of white pixels that belong to a specific concavity configuration. The label for each white pixel is chosen based on the Freeman code with four directions. Each direction is explored up to an encounter with a black pixel or until the limits imposed by the digit bounding box are met. A white pixel is labeled if at least two consecutive directions find black pixels (see Figure 5-12 for the digit 3).



Feature vector computed for the 1st column

0	3	29	0	0	0	0	0	0	0	0	0	0
1	2	3	4	5	6	7	8	9	A	B	C	D

Code Z is ignored.

Figure 5-12 Concavity features calculated for the digit 3

Thus, we have 9 possible concavity configurations (see Figure 5-13). Moreover, we consider four more configurations, as defined in [HEUTE et al., 1998], in order to detect more precisely the presence of loops. The total length of this feature vector is then 13. The concavity vector is normalized between 0 and 1, by the total of the concavity codes computed for each column or row of the digit image.

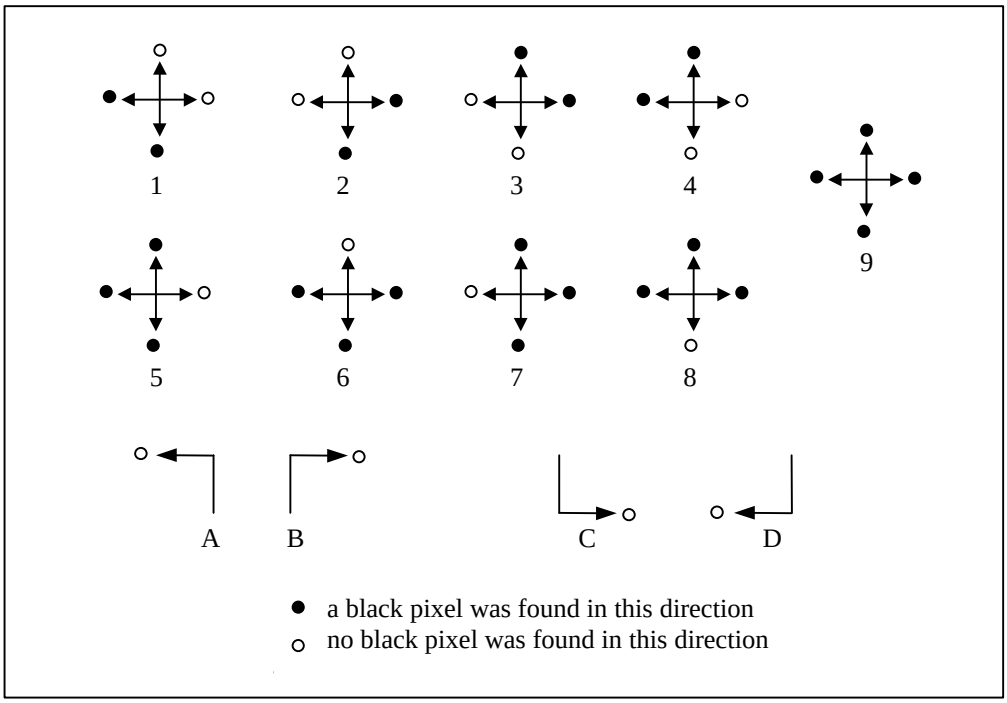


Figure 5-13 Concavity configurations used for labelling white pixels

5.3.2. Verification module

The verification module is based on Viterbi's Algorithm described in Chapter 3. Each digit class is represented by two numeral HMMs: one based on the image columns ($l_c^0, l_c^1, ..., l_c^9$) and other based on the image rows ($l_r^0, l_r^1, ..., l_r^9$) of the digit images.

In this module, the first step concerns the selection of the column ($l_{vc}^0, l_{vc}^1, ..., l_{vc}^9$) and row ($l_{vr}^0, l_{vr}^1, ..., l_{vr}^9$) models corresponding to the string segment to be verified. This is done using the segment recognition result provided by the SCB stage. The selected pair of models is scored using Viterbi's algorithm and the sequence of observations extracted by the FBFE module. The output probabilities of both the column and row models are combined by summing their logs. The resulting probability corresponds to the probability of the string segment being correct. A string probability can be obtained by summing the log of the probability of each segment of the numeral string. The string probability calculated in the Verification module is added to the probability of the segmentation-recognition hypothesis obtained from the SCB stage. The final resulting probability is used to re-rank the string segmentation-recognition hypothesis.

To better explain, let us to consider the *N* segmentation-recognition hypotheses provided by the SCB stage as $(srh_1,...,srh_N)$, where each srh_i is a structure composed of the following fields: string length(M), segmentation points $(sp_{i1},...sp_{iM-1})$, recognition result of each string segment$(r_{i1},...,r_{iM})$ and the probability of the segmentation-recognition hypothesis$(Psrh_i)$ being correct. In the Verification stage, the first step is carried out by the FBFE module, in which the segmentation points$(sp_{i1},...sp_{iM-1})$ are used to define each segment seg_{ij} and its bounding box. A feature vector is extracted from each column and row of seg_{ij} taking into account its bounding box. The column and row based feature vectors are quantized in order to generate two sequences of discrete observations representing each seg_{ij} (see Figure 5-14), respectively. The Verification module uses the recognition result r_{ij} of the corresponding segment to select the column- and row-based numeral HMMs, $l_c^{r_{ij}}$ and $l_r^{r_{ij}}$, which are scored using the respective sequence of observations. A segment recognition probability$(Pseg_{ij})$ is estimated by summing the probability of the column and row numeral models. The global string probability is obtained by:

$$Pstr_i = \sum_{j=1}^{M} \log Pseg_{ij} \tag{5.8}$$

Finally, the segmentation-recognition probability of the SCB stage $(Psrh_i)$ and that obtained in the Verification stage $(Pstr_i)$ are combined as:

$$Pstring_i = \log Psrh_i + \log Pstr_i . \tag{5.9}$$

The resulting string probability $(Pstring_i)$ is used to re-rank the ith string recognition hypothesis.

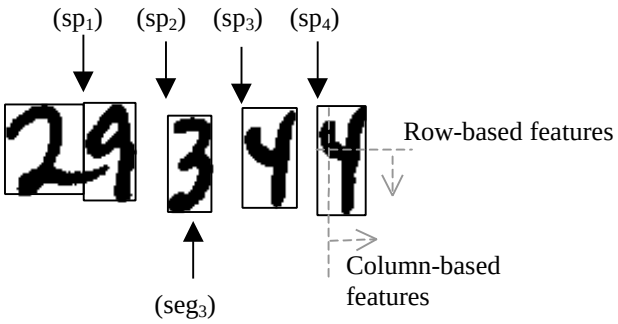


Figure 5-14 A numeral string, the segmentation points (sp_i), a segment and the corresponding bounding box (seg_3) and the column- and row-based feature extraction

5.3.3. Numeral models

The digit classes are represented by 20 models: 10 column-based (l_c^0, l_c^1,..., l_c^9) and 10 row-based (l_r^0, l_r^1,..., l_r^9) models. They have the same end-state topology used for the numeral models of the SR module. The optimization scheme defined in Chapter 3 is also used to define the final length (number of states) of these models (see Table 5-3). However, they differ from the SR models in the sense that they are trained without considering any string contextual information and using a feature set based on both foreground and background information. Moreover, there is no space model inside them. The objective is to obtain numeral models which are more powerful in terms of isolated digit recognition performance than those used in the SR module.

Table 5.3 Number of states of the numeral models

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

5.4. Summary

In this chapter, the proposed method was described in detail. To deal with the tradeoff between segmentation and recognition a two-stage recognition method was proposed, which can be categorized as a segmentation-free approach. In the first stage, a prior segmentation of strings into digits is avoided through the use of an implicit segmentation strategy. The second stage was used as a verification step. Within a general overview of the method, each stage and its modules were described. The String Context-Based (SCB) Stage and its modules: Preprocessing, Foreground Feature Extraction (FF) and Segmentation-Recognition (SR) and the Verification Stage were detailed. We have described the HMM models used to represent the digit classes, and an evaluation of different HMM topologies in the LBA framework. The use of an end-state in the HMM topology allowed a better distribution of the observations among their states. This has brought an improvement in terms of segmentation performance in the SCB stage. This additional state in the HMM topology has also enabled to incorporate contextual knowledge regarding strings into the numeral models used in the SCB stage. To this purpose, we have used a string-based training to build a space model inside of the numeral models. Finally, we have described the Verification stage used to re-rank the best hypotheses generated by the SCB stage. A set of background features is used as complementary information for the foreground features used in the SCB stage.

6. Experimental Results

In this chapter, experiments undertaken during the course of development of the proposed method are detailed. Section 6.1 shows that each experiment is performed considering isolated digits and numeral strings of different lengths extracted from NIST SD19 database. A detailed description of this database is available in [GROTHER, 1995].

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

The protocol used to implement and evaluate the proposed method consists of three steps. Section 6.2 presents the first step, which is called SCB Stage Construction and Evaluation. In this step, we construct a baseline system composed of the Preprocessing, Forward Features Extraction (FFE) and Segmentation-Recognition (SR) modules representing the first stage of the proposed numeral string recognition method. The Preprocessing module is constrcuted taking into account the preliminary analyses described in Chapter 4. The performance of the SCB stage was improved by adding an end-state to the HMM structure. This end-state contributed to improving the segmentation performance of the LBA. Moreover, a space model was built into the numeral models. Finally, further improvement was obtained by optimizing the HMM parameters using the scheme described in [WANG, 1994].

Section 6.3 corresponds to the second step of the evaluation protocol, called Verification Stage Construction and Evaluation. The objective is to re-rank the N best segmentation-recognition hypotheses of the SCB stage by using a verification strategy. For this purpose, an isolated digit classifier is developed to check each string segment in the segmentation-recognition hypotheses of the SCB stage, the general idea here being

to evaluate the SCB stage with respect to segmentation and recognition of numeral strings, and to include a further verification step to check and re-rank its results.

In the last step of the evaluation protocol the system is evaluated using a non-informed strategy, where the string length is unknown. An error analysis is also presented.

6.1. Databases

The isolated numerals used in these experiments come from NIST SD19, we use 50,000 numeral samples for training, 10,000 for validation and 10,000 for testing. The training samples were extracted from *hsf_0, hsf_1 and hsf_2*, the validation samples from *hsf_7* and the testing samples from *hsf_4*.

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

6.2. SCB stage - construction and evaluation

This section corresponds to the first step of the evaluation protocol, in which the SCB stage is improved at each new experiment. This stage is composed of Preprocessing, FFE and SR modules as shown in the general overview presented in Chapter 5.

In the experiments on string normalization, both slant normalization techniques proposed in Chapter 4, with and without contextual information, are developed and evaluated. Moreover, the advantage of using contextual information instead of a size normalization method to deal with the intra-string size variation is shown. These experiments also show that the foreground features proposed in the FFE module are really unaffected by inter-string size variation.

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

6.2.1. Baseline system

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

6.2.2. Experiments on slant normalization

These experiments are designed to answer the third question proposed in Chapter 4: What is the real contribution to numeral string recognition achieved by string slant to normalize the isolated digits used to train the numeral models of the SCB stage? To this end, the baseline system is used to compare recognition performance by considering no slant normalization, slant normalization without contextual information and with contextual information (CI). In the experiment based on slant normalization without contextual information, each isolated numeral used for training the numeral HMMs is slant-corrected using the slope estimated from its own image (q_1 in Figure 4-9). In contrast, when contextual information is used, the slope for each isolated numeral is estimated as the slope calculated for its original string (q_2 in Figure 4-9).

Table 6.1 Slant normalization experiments - isolated digit recognition

Experiments	Validation (%)	Testing (%)
Without slant normalization	93.1	86.2
Slant normalization without CI	95.9	93.0
Slant normalization with CI	95.8	92.6

Table 6-1 presents some preliminary recognition results for isolated digits. We can observe that contextual information does not contribute to the recognition of slant

normalized isolated numerals, since the numeral origin is not helpful information in this case.

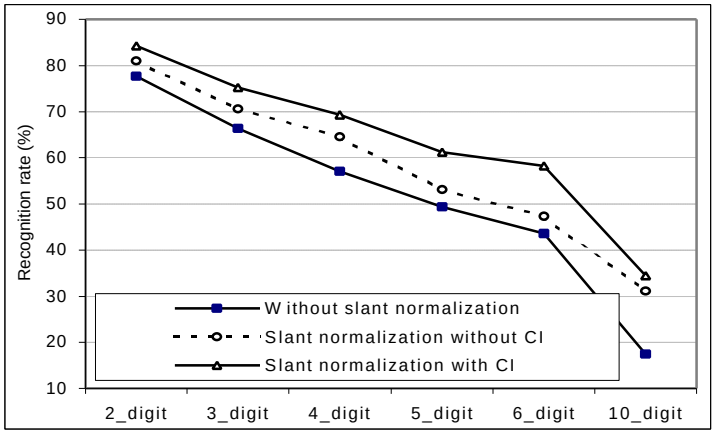


Figure 6-1 Slant normalization experiments - string recognition results

Table 6.2 Slant normalization experiments - string recognition results

Experiments	Without slant normalization (%)	Slant normalization without CI (%)	Slant normalization with CI (%)
2_digit	77.63	80.97	84.21
3_digit	66.37	70.64	75.17
4_digit	57.01	64.47	69.25
5_digit	49.30	53.10	61.18
6_digit	43.56	47.34	58.18
10_digit	17.42	31.05	34.42
Global	55.18	60.58	66.48

On the other hand, Figure 6-1 and Table 6-2 show interesting results for numeral strings. Normalizing without contextual information has brought an improvement of 5.4% in the global string recognition rate, while the use of contextual information allows an improvement of 11.3%, both compared to the experiments without slant normalization.

6.2.3. Experiments on size normalization

The preliminary analysis presented in Chapter 4 has shown a significant intra-string size variation in the NIST database. In these experiments, we evaluate two strategies to deal with this problem: a) the use of the non-linear size normalization method described in Chapter 4 to normalize each numeral string along the vertical axis by using the mean digit height (45 pixels) calculated from the training database; b) the

use of contextual information regarding intra-string size variation (Intra-SSV) during training of the numeral models. In the last strategy, features from each training sample representing an isolated digit are extracted taking into account the height of its original string bounding box instead of the height of its own bounding box (see Figure 6-2). Moreover, we consider the foreground features as size invariant. This means that nothing is done to deal with inter-string size variation.

Figure 6-2. a) Original string; b) String bounding box after slant normalization; c) Training samples linked to their orginal strings and the bounding box used for feature extraction.

Table 6.3 Size normalization experiments - isolated digit recognition

Experiments	Validation (%)	Testing (%)
Without size normalization	95.80	92.60
Size normalization	95.23	92.20
Intra-SSV	94.85	91.10

Table 6.4 Size normalization experiments - string recognition results

Experiments	Without size normalization (%)	Size normalization (%)	Intra-SSV (%)
2_digit	84.21	84.23	85.32
3_digit	75.17	75.68	78.19
4_digit	69.25	69.97	71.34
5_digit	61.18	64.03	66.32
6_digit	58.18	59.79	63.85
10_digit	34.42	40.75	44.04
Global	66.48	68.08	70.43

Table 6-3 shows that the foreground features really do not vary with inter-string size variation. Moreover, the experiment based on Intra-SSV loses in terms of numeral recognition, which is possible since the use of this additional contextual information increases numeral variability. In contrast, this experiment brought a further improvement of 3.95% to the global string recognition rate (see Table 6-4), even with a small loss in terms of isolated numeral recognition performance.

The experiments considering the use of a size normalization method have shown some improvement, but not so relevant as the one brought by the use of Intra-SSV for

training the numeral models. This is due to additional touching digits and distortions in the digit strokes caused by the size normalization method.

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

These experiments show that the HMM topology with end-state does not bring about a significant improvement in the recognition of isolated numerals (about 0.75% in Table 6-5). On the other hand, it brought about a 7.08% improvement in the global string recognition rate. This is due the better distribution of the observations among the HMM states, and a better estimation of the self-transition probability in the last HMM state (s_5). Consequently, the LBA provides a more precise match of numeral models to the observation sequence. This means a better definition of string segmentation points.

Table 6.5 End-state experiments - string recognition results

Class	HMM without end-state (%)	HMM with end-state (%)
2_digit	85.32	87.72
3_digit	78.19	82.43
4_digit	71.34	78.17
5_digit	66.32	75.65
6_digit	63.85	71.69
10_digit	44.04	60.64
Global (All classes)	**70.43**	**77.51**

Figure 6-3 shows an example in which the segmentation cuts at the top and bottom were provided respectively by the models with and without an end-state. To confirm the improvement to segmentation cuts, we carried out an error analysis considering the 10_digit strings misrecognized using the models without an end-state, which were all recognized with the models with an end-state. A total of 245 samples were manually checked.

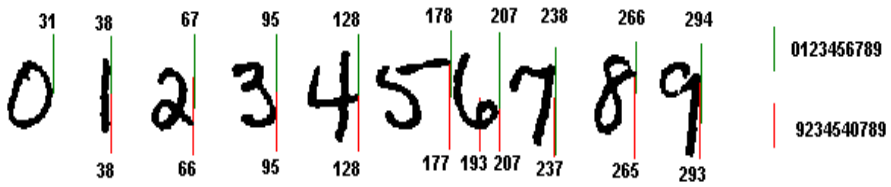


Figure 6-3 Segmentation points and recognition result produced by the LBA using 5-state HMMs with an end-state (top) and without an end-state (bottom).

Figure 6-4 shows that 72.2% of these misrecognitions are related to mis-segmentation problems. Moreover, we compute the difference of location of the

segmentation points provided by the two HMM structures in terms of the number of observations. Figure 6-5 shows that the frequencies of location differences equal to 1, 2 and more than 2 observations are, respectively, 47.3%, 18% and 6.9%.

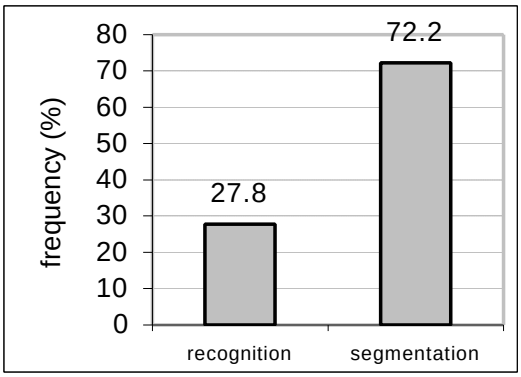


Figure 6-4 Frequency of recognition and segmentation mistakes

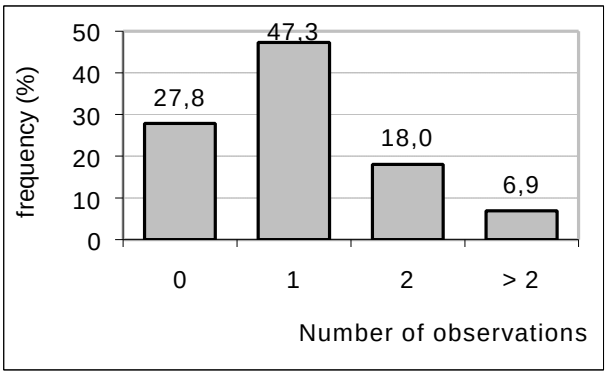


Figure 6-5 Difference between the location of segmentation points considering the number of observations

6.2.5. Contribution of a space model

In these experiments, the digit pair database and the two-step training mechanism described in Chapter 5 are used to evaluate the use of a space model. This corresponds to an additional state in the numeral HMM structure. In these experiments, the following strategies are evaluated: 1) the use of one space model for each numeral class; and 2) the use of one space model representing all numeral classes. In both strategies, the numeral models are first trained on isolated digits. Subsequently, the space model parameters are estimated during the second-step training on digit pairs. The parameters corresponding to the numeral models are kept the same as estimated during the first training step on isolated numerals.

Table 6-6 summarizes all these experiments. It can be observed that the space model brings about some improvement for each string class. The recognition

performance of both strategies is almost the same, which shows that the space model is not dependent on digit class.

Table 6.6 Space model experiments – isolated digits and string recognition

Class	HMM with end-state (%)	HMM with end-state and space model* (%)	HMM with end-state and space model** (%)
2_digit	87.72	87.93	87.93
3_digit	82.43	82.73	82.60
4_digit	78.17	78.46	78.42
5_digit	75.65	76.03	76.12
6_digit	71.69	72.00	72.01
10_digit	60.64	61.13	61.30
Global (All classes)	**77.51**	**77.83**	**77.83**

(* one space model by digit class; ** one space model for all numeral classes)

6.2.6. Optimization of the HMM parameters

The scheme described in Chapter 5 for defining HMM length is used to redefine the number of states of the numeral HMMs in the SCB stage. In addition, a new codebook is evaluated. The best results are achieved by using the minimum values shown in Table 6-7, and a codebook with 256 entries. Table 6-8 shows the impact on the recognition performance for isolated digits, while Table 6-9 does so for numeral strings. In these experiments, we did not consider the space model.

Table 6.7 Minimum, mean and maximum length for each numeral HMM

Numeral model	Number of states		
	Min	Mean	Max
0	13	18	24
1	5	6	7
2	14	22	30
3	14	20	26
4	15	22	28
5	13	21	29
6	15	20	25
7	15	20	25
8	14	17	24
9	16	20	25

Table 6.8 Different HMM configurations – isolated digit recognition

HMM based on Foreground features	Validation (%)	Testing (%)
Baseline system	95.60	91.73
Optimized HMMs and 256-codebook	96.79	94.00

Table 6.9 HMM parameters optimization – string recognition results

Class	Baseline system (%)	Optimized system (%)
2_digit	87.72	89.83
3_digit	82.43	84.49
4_digit	78.17	80.38
5_digit	75.65	78.06
6_digit	71.69	75.84
10_digit	60.64	66.23
Global (All classes)	77.51	80.36

We update the space model experiments considering the new HMM parameters. One space model representing all numeral classes is considered in Table 6-10.

Table 6.10 String recognition results after optimizing the HMM parameters and using a space model

Class	System without space model	System with space model
2_digit	89.83	90.29
3_digit	84.49	85.87
4_digit	80.38	81.66
5_digit	78.06	79.97
6_digit	75.84	76.76
10_digit	66.23	68.44
Global (All classes)	80.36	81.65

6.3. Verification stage - construction and evaluation

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

The same scheme used for optimizing the numeral HMMs of the SR module is applied to define the length of these new HMM models. The gap between the number of states in the baseline system and those estimated using the scheme proposed in Chapter 4 is very large (see Table 6-11). For this reason, we decide at this time to evaluate, for the column-based models, configurations with 6, 8 and 12 states. Table 6-12 shows the recognition results considering different number of states for the column and row numeral models.

Table 6.11 Minimum, mean and maximum length for each numeral HMM

	Column based models			Row based models		
Numeral model	Min	Mean	Max	Min	Mean	Max
0	13	18	24	14	21	28
1	5	6	7	16	24	32
2	14	22	30	16	24	32
3	14	20	26	20	28	36
4	15	22	28	18	28	39
5	13	21	29	19	27	35
6	15	20	25	18	27	36
7	15	20	25	18	27	36
8	14	17	24	20	29	38
9	16	20	25	21	31	41

Table 6.12 Experiments on isolated digits considering different number of states in the numeral HMMs

	Column models		Row models	
Number of states	Validation (%)	Testing (%)	Validation (%)	Testing (%)
6	97.63	94.55	95.65	92.27
8	97.78	94.90	-	-
12	97.89	95.26	-	-
Minimum values (Table 6-11)	98.01	95.51	97.56	95.16
Mean values (Table 6-11)	97.54	94.61	97.40	95.02

The best result is obtained by using the minimum values in Table 6-11. The maximum values are not evaluated since we have observed a loss in terms of recognition rates for the mean values. The codebook size is experimentally optimized. We evaluate codebooks composed of 64, 128, 192, 256 and 320 entries. The codebook composed of 256 entries provided the best results (see Table 6-13).

Table 6.13 Experiments based on different codebook sizes – isolated digit recognition

	Column models		Row models	
Codebook size	Validation (%)	Testing (%)	Validation (%)	Testing (%)
64	95.40	92.94	-	-
128	97.89	95.26	97.56	95.16
192	98.24	96.23	98.16	96.63
256	98.44	96.54	98.40	97.09
320	98.32	96.44	98.30	96.92

Finally, Table 6-14 shows the recognition results for isolated digits when the column and row models are combined. They are combined by summing the log of the final probability of each model calculated using Viterbi's algorithm.

Table 6.14 Combination of column and row models – isolated digit recognition

	Validation (%)	Testing (%)
Column based features	98.44	96.54
Row based models	98.40	97.09
Combination (column x row models)	99.00	98.02

6.4. Recognition results of known length strings

During these experiments, the SR module provides the 10 best segmentation-recognition paths for each numeral string. In the Verification stage, the FBFE module uses the segmentation points of each path as delimiters in the preprocessed string image to calculate new features based on columns and rows for each string segment. The recognition result of the first stage is verified using the new set of features and numeral HMMs available in the Verification stage. We combine the recognition results of the SCB and Verification stages, as described in Chapter 5. Table 6-15 shows the top 5 recognition results of the first stage of our system, while Table 6-16 presents the top 5 recognition results after the Verification stage. The last line of these tables shows the recognition results for Touching Digit Pairs (TDPs) using a database composed of 2,069 samples extracted from the NIST database.

Table 6.15 SCB stage – numeral string recognition

Class	Top (1)	Top (2)	Top (3)	Top (4)	Top (5)
2_digit	90.29	95.35	96.91	97.25	97.46
3_digit	85.87	91.99	92.83	93.20	93.33
4_digit	81.66	89.38	91.17	91.81	91.98
5_digit	79.97	87.69	89.55	90.50	90.67
6_digit	76.76	85.85	87.32	88.47	88.84
10_digit	68.44	73.62	74.28	74.44	74.44
Global	81.65	88.57	90.00	90.62	90.81
TDPs	79.51	88.44	91.64	92.65	93.19

Table 6.16 SCB + Verification stage numeral string recognition

Class	Top (1)	Top (2)	Top (3)	Top (4)	Top (5)
2_digit	95.23	97.59	98.35	98.48	98.57
3_digit	92.62	95.60	96.18	96.27	96.28
4_digit	92.11	95.35	95.95	96.03	96.12
5_digit	90.00	93.96	94.52	94.69	94.73
6_digit	90.09	94.05	94.88	94.92	95.02
10_digit	86.94	90.30	90.38	90.46	90.46
Global	91.57	94.86	95.47	95.57	95.63
TDPs	89.61	94.39	95.36	95.70	95.84

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

6.5. Recognition results of unknown length strings

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

To deal with this problem, a maximum value is assigned to the *L* parameter of the LBA. This parameter defines the number of levels to be built in the algorithm. Each level corresponds to a digit in the numeral string. Since the larger numeral string in the NIST database is composed of 10 digits, we decide to define L=12. Table 6-17 shows the string recognition results, where it is possible to observe a small loss in terms of recognition performance.

Table 6.17 SCB + Verification stage – recognition of unknown-length strings

Class	Top (1)	Top (2)	Top (3)	Top (4)	Top (5)
2_digit	94.13	96.50	97.21	97.22	97.22
3_digit	91.49	94.30	94.72	94.75	94.76
4_digit	91.00	93.94	94.50	94.58	94.63
5_digit	88.43	92.36	92.83	92.96	93.00
6_digit	89.58	93.31	94.05	94.10	94.19
10_digit	85.87	88.99	89.07	89.15	89.15
Global	90.48	93.62	94.14	94.20	94.24
TDPs	86.76	91.01	91.88	92.12	92.41

In another experiment, the use of the string length predictor based on Bayes theory described in Chapter 3 is considered. It uses the minimum-error-rate decision rule to predict the string length (number of digits) given the width of the string bounding box (*sbb*) in terms of number of columns. A set of string classes is defined as

$w = \{2\_digit, 3\_digit, 4\_digit, 5\_digit, 6\_digit, 10\_digit\}$, in which class $\#\_digit$ corresponds to strings composed of # digits. The a priori probabilities of these classes are considered ambiguous, *i.e.* $P(2\_digit) = P(3\_digit) = P(4\_digit) = P(5\_digit) = P(6\_digit) = P(10\_digit)$. The parameters of a Gaussian *pdf* are estimated for each class by using a training set composed of 44,256 handwritten numeral strings extracted from the NIST SD19 database. Figure 6-6 shows the number of samples per string bounding box width, while Figure 6-7 presents the a priori probability of each *sbb* width for each string class.

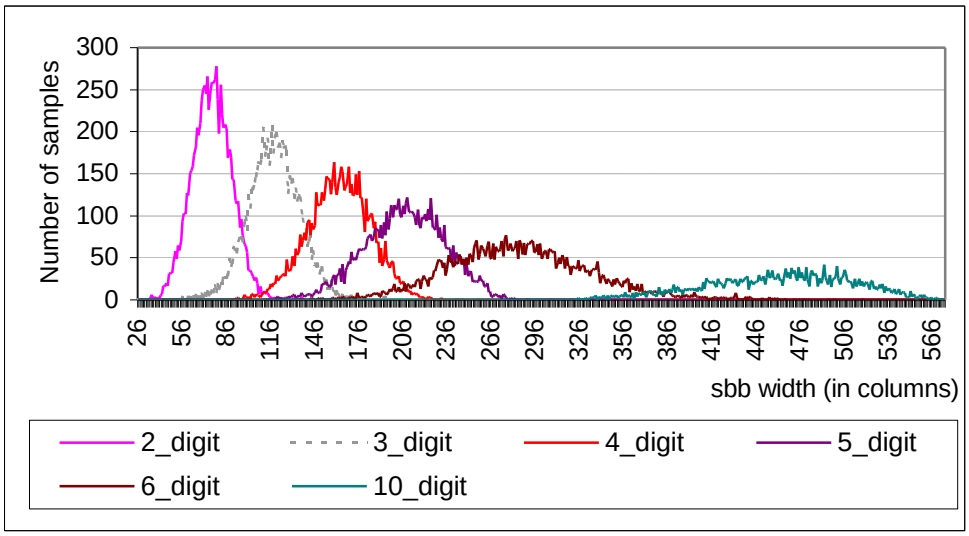


Figure 6-6 Number of samples per string bounding box width for each class

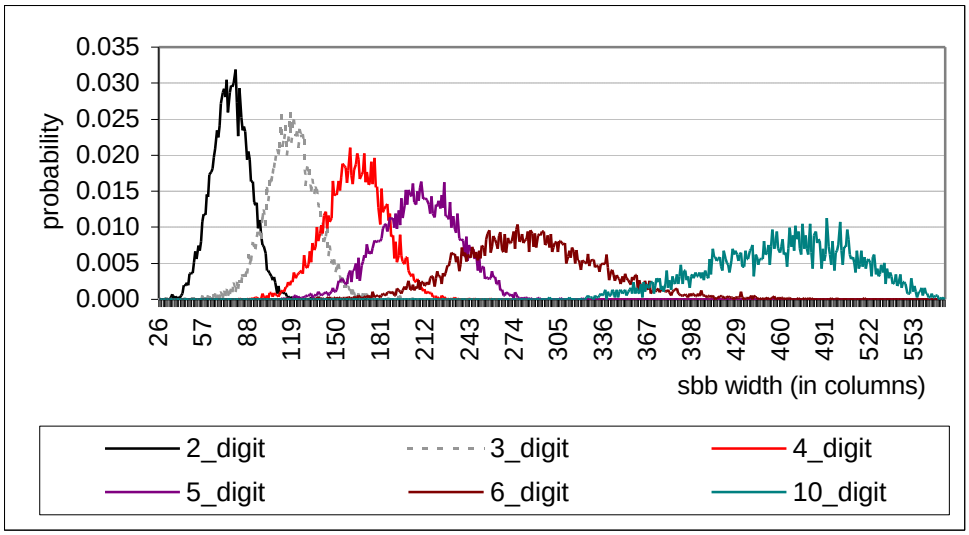


Figure 6-7 A priori probability of the sbb width for each string class

Then, a string length classifier is designed to classify the *sbb_width* into M classes of string lengths by using M discriminant functions $g_j(sbb\_width)$, computing the similarities between the unknown data *sbb_width* and each string class w_j and selecting the class w_i corresponding to

$$g_i(sbb\_width) > g_j(sbb\_width) \qquad for\ all\ \ j \neq i\,. \tag{6.1}$$

Figure 6-8 shows the scheme of this classifier, where the decision rule is to maximize the a posteriori probability described in Equation 3.60.

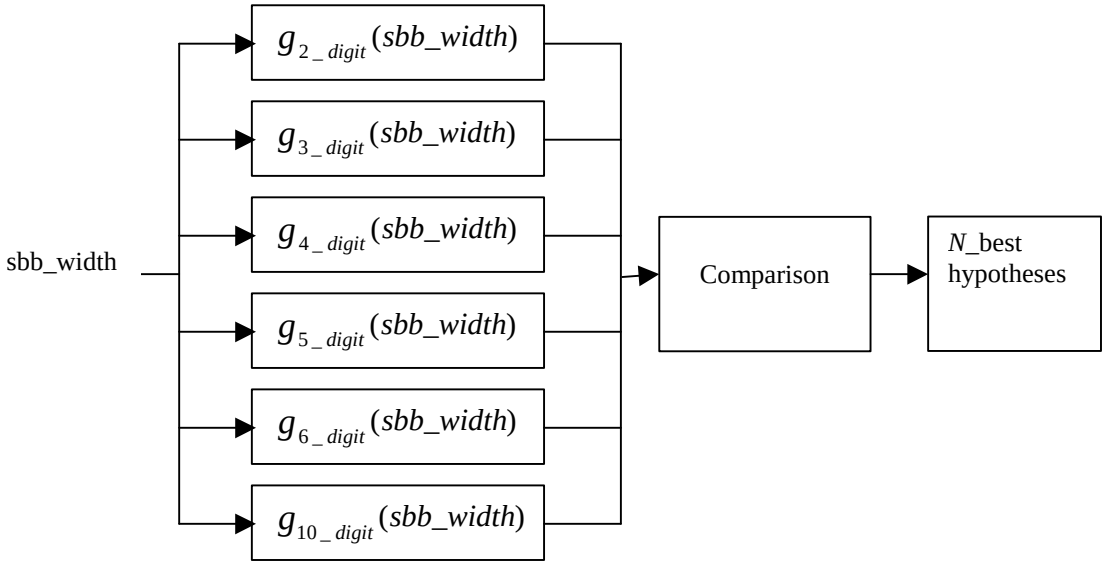


Figure 6-8 Scheme of the classifier used to predict the string length from the width of the string bounding box (ssb_width)

The same testing set composed of 12,802 numeral strings used to evaluate the recognition method is used to test this string length predictor. Figure 6-9 shows the classification results. It is possible to observe that when the right decision is considered in the best 3 hypotheses, the performance of the string length predictor is very promising.

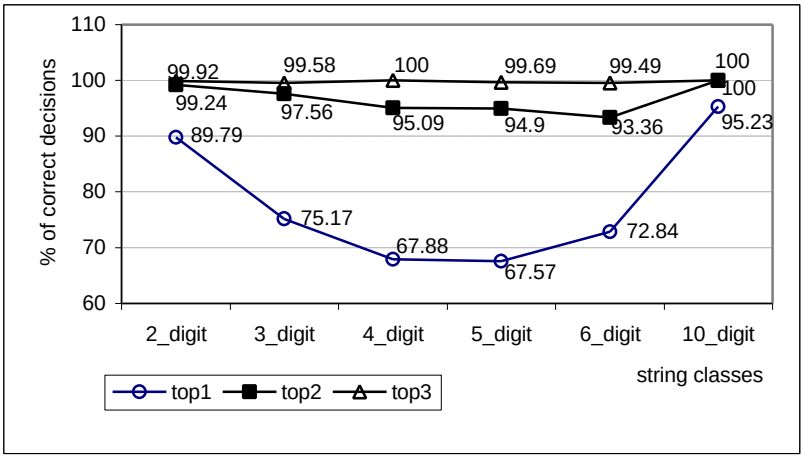


Figure 6-9 Classification results of the *sbb_width* into string length classes using the proposed classifier

Thus, the top 3 decisions of the string length predictor are used to determine the *L* parameter of the LBA instead of fixing it at 12. For instance, if the top 3 decisions belong to the 3*_digit*, 4*_digit* and 5*_digit* class, then five levels are constructed by the LBA and after that the probabilities of each segmentation-recognition hypothesis corresponding to these string lengths are compared. The segmentation-recognition hypothesis with the highest probability is chosen. Table 6-18 shows the recognition results when the string length is predicted using the proposed string length classifier.

Table 6.18 SCB + Verification stage – recognition of unknown length strings

Class	Top (1)	Top (2)	Top (3)	Top (4)	Top (5)
2_digit	94.81	97.17	97.93	98.05	98.14
3_digit	91.61	94.61	95.05	95.09	95.09
4_digit	91.25	94.29	94.84	94.93	94.97
5_digit	88.30	92.18	92.66	92.79	92.83
6_digit	89.07	92.81	93.55	93.59	93.68
10_digit	86.94	90.30	90.38	90.46	90.46
Global	90.66	93.87	94.41	94.50	94.54
TDPs	88.98	93.57	94.88	95.36	95.70

6.6. Error analysis

Table 6-19 shows the confusion matrix computed from the isolated digit recognition results of the SCB stage. This matrix confirms that the numeral models of the SCB stage are not powerful enough in terms of recognition performance of isolated digits. Their weakness are related to the feature extraction method used in this stage, the objective of which is to maximize the likelihood of the segmentation and recognition of numeral strings in an implicit segmenation-based process. Most of the confusion occurs

between digit classes: 0-6, 5-3, 7-2 and 9-4. However, we observed that the SCB stage is often able to find the right segmentation points for a given string, even without achieving the right recognition. This is possible, given the similar length of the observation sequences representing the digit classes involved in these confusing situations.

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

Table 6.19 Confusion matrix – isolated digit recognition of the SCB Stage

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

The confusion matrix in Table 6-20 shows that combining column and row numeral models to represent each digit class provides an interesting recognition performance. The background features based on concavities have shown to be a promising way to distinguish between classes 0-6, 5-3, 7-2 and 9-4. However, there is still some confusions between classes 5-8, 5-9, 6-5, 7-2 and 9-4.

Table 6.20 Confusion matrix – isolated digit recognition of the verification Stage

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

Table 6-21 shows the system mistakes related to the recognition of handwritten numeral strings categorized as: a) segmentation caused by touching problems; b) segmentation caused by overlapping problems; c) digit recognition; and d) presence of noise in the string images. Most of the time, the segmentation mistakes are related to touching or overlapping problems. As expected the worst segmentation problem concerns touching digits (21.28%). Few segmentation mistakes (0.85%) occurred in naturally segmented strings and they are due to the lack of samples for training the space model. The digit pair database is not representative of strings of different lengths. However, most of the mistakes in the applying the method are due to recognition problems (62.12%). Few mistakes are related to presence of noise in the string images.

Table 6.21 Summary of the system mistakes (%)

System errors in percentage			
Segmentation	Touching	21.28	
	Overlapping	12.34	
	Naturally segmented	0.85	33.62
Recognition			62.12
Noise			4.26

Table 6-22 shows some examples of incorrectly recognized strings, while Table 6-23 presents some examples of correctly recognized strings.

Table 6.22 Examples of incorrectly recognized strings

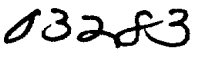
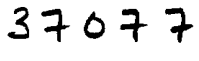
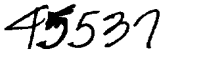
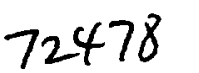
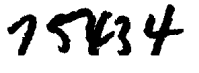
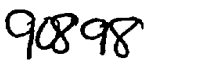
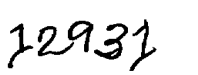
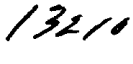
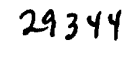
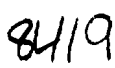
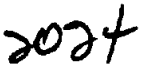
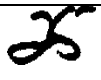
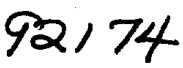
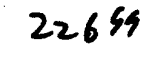
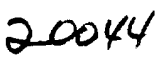
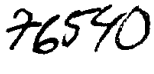
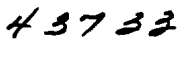
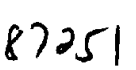
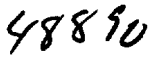
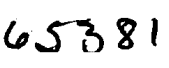
 84187 (84297)	 03223 (03283)
 32072 (37077)	 95537 (45537)
 47534 (97539)	 60114 (60118)
 77478 (72478)	 78434 (75434)
 90398 (90898)	 72937 (12931)
 88 (86)	 68170 (60170)
 13211 (13210)	 22344 (29344)

Table 6.23 Examples of correctly recognized strings

 54593	 8419
 2024	 6001
 7433	 3258
 25	 50
 56	 25
 92174	 22699
 20044	 69878
 20356	 76540
 43733	 87251
 48890	 65381

6.7. Discussion

From the preliminary analyses on the NIST database presented in Chapter 4, we have observed that the use of a slant normalization method brought about a significant reduction in the number of overlaps between adjacent numerals in strings. The proposed MKSC slant normalization method achieves a more significant reduction (60.83%) than that provided by the KSC method (49.42%). This has indicated that slant normalization is really helpful in alleviating overlapping problems. Moreover, slant estimated from isolated digits and that estimated from their original strings differ by more than 10.95^{o} in 11.74% of the 197,784 digits analyzed using the MKSC method. The proposed

MKSC method also reduces the number of cases where this difference is significant. This means that we can approximate the slant estimated from the string to the slant estimated from each isolated numeral in the string. All these observations justify an investigation of the real contribution to string recognition of using the proposed slant normalization based on contextual information. This is achieved through the experiments presented in Section 6.2.2. They have shown that the use of contextual information to provide the same conditions during training and testing is a promising strategy in an implicit segmentation-based method. The use of slant normalization brought about an improvement of 5.4% to the global string recognition rate, while the use of slant normalization based on contextual information allowed an improvement of 11.3%.

The strategies used to deal with size variation have also taken into account the implicit segmentation-based method of the SCB stage. The non-linear size normalization method described in Chapter 4 has shown to be a promising way to reduce the intra-string size variation. However, the experiments in Section 6.2.3 have shown that training the numeral HMMs taking into account the intra-string size variation is more promising. With this contextual information during training of the numeral models, we avoid the possible distortions on the digit strokes caused by a non-linear size normalization method. These experiments have shown that the use of size normalization brought about an improvement of 1.6% in recognition performance, while the use of IntraSSV as contextual information brought about an improvement of 3.95%.

The HMM topology with an end-state has ensured a more precise definition of the string segmentation cuts by the LBA. The experiments presented in Section 6.2.4 shows that although it does not bring about a significant improvement in the recognition of isolated numerals (about 0.75% in Table 6-5), it did bring about an improvement of 7.08% in the global string recognition rate. This additional state also enabled the use of a two-step training mechanism to incorporate further string contextual information in the numeral models. Section 6.2.5 shows experiments considering the use of a space model built inside of the numeral models. We can observe that the space model brought about some improvement in terms of recognition performance for each string class. Even with not such significant results, these experiments have shown that an investigation on modeling other kinds of interactions between adjacent digits, such as

touching and overlapping, may be a promising way of obtaining accurate string representation. To this end, further work can be done to train a pause model instead of a space model.

The scheme used for optimizing the HMM parameters of the numeral models in the SCB stage provides a further improvement of 2.81% in the global string recognition rate. In this optimization scheme, we have redefined the number of states of the numeral HMMs and also the codebook size.

After all these experiments had been conducted to construct and evaluate the SCB stage, the final recognition rates for strings composed of 2, 3, 4 5 6 and 10 digits were: 90.29%, 85.87%, 81.66%, 79.97%, 76.76% and 68.44% respectively.

The Verification stage has been shown to make a significant contribution to string recognition performance. After this stage, the final recognition rate of known length strings composed of 2, 3, 4, 5, 6, and 10 digits were: 95.23%, 92.62%, 92.11%, 90.00%, 90.09%, and 86.94% respectively. This means an improvement on the global string recognition rate of 9.92%. Similarly, the recognition rate of touching digit pairs (TDPs) had improved from 79.51% to 89.61%.

The strategies used to consider unknown-length numeral strings showed a small loss in terms of recognition performance compared to the previous results. The most promising results were obtained using the string length predictor. The final recognition rates of strings composed of 2, 3, 4, 5, 6, and 10 digits were: 94.81%, 91.61%, 91.25%, 88.30%, 89.07%, and 86.94% respectively. This means a loss in terms of global recognition rate of 0.91%.

Table 6-24 shows a comparison of the proposed method with other methods. Even considering only methods evaluated on numeral strings extracted from the NIST database, the comparison with other methods is delicate in some cases because of the uncertainty concerning the exact data being used and the different number of samples. Table 6-25 presents a similar comparison for touching digit pairs.

Table 6.24 Performance of Numeral Strings based data in NIST SD19

Reference	String class	Recognition rate	Error rate	Rejection rate	# of samples on the testing set
[KEELER & RUMELHART 1992]	2_digit	87.00	1.00	12.00	1,000
	3_digit	84.00	1.00	15.00	1,000
	4_digit	76.00	1.00	23.00	1,000
	5_digit	71.00	1.00	28.00	1,000
	6_digit	62.00	1.00	37.00	1,000
	10_digit	NA	NA	NA	NA
[FUJISAWA & NAKANO 1992]	2_digit	89.79	10.21	NA	1,000
	3_digit	84.64	15.36	NA	1,000
	4_digit	80.63	19.37	NA	1,000
	5_digit	76.05	23.95	NA	1,000
	6_digit	74.54	25.46	NA	1,000
	10_digit	NA	NA	NA	NA
[MATIN et al., 1993]	2_digit	94.20	1.00	4.80	1,000
	3_digit	87.90	1.00	11.10	1,000
	4_digit	79.90	1.00	19.10	1,000
	5_digit	75.60	1.00	23.40	1,000
	6_digit	63.30	1.00	35.70	1,000
	10_digit	NA	NA	NA	NA
[HA et al., 1998]	2_digit	96.20	3.80	0.00	981
	3_digit	92.70	7.30	0.00	986
	4_digit	93.20	6.80	0.00	988
	5_digit	91.10	8.90	0.00	988
	6_digit	90.30	9.70	0.00	982
	10_digit	NA	NA	NA	NA
[LEE & KIM, 1999]	2_digit	95.23	4.77	NA	1,000
	3_digit	88.01	11.99	NA	1,000
	4_digit	80.69	19.31	NA	1,000
	5_digit	78.61	21.39	NA	1,000
	6_digit	70.49	29.51	NA	1,000
	10_digit	NA	NA	NA	NA
[ELMS et al., 1998]	2,3,4,5,6 and 10_digits	74.20	25.80	NA	1,400
[YOON et al., 2000]	2_digit	92.00	8.00	0.00	24
	3_digit	100.00	0.00	0.00	21
	4_digit	95.00	5.00	0.00	21
	5_digit	94.00	6.00	0.00	17
	6_digit	92.00	8.00	0.00	12
	10_digit	100.00	0.00	0.00	5
Proposed method (known length strings)	2_digit	95.23	4.77	0.00	2,370
	3_digit	92.62	7.38	0.00	2,385
	4_digit	92.11	7.89	0.00	2,345
	5_digit	90.00	10.00	0.00	2,316
	6_digit	90.09	9.91	0.00	2,169
	10_digit	86.94	13.06	0.00	1,217
Proposed method (unknown length strings)	2_digit	94.81	5.19	0.00	2,370
	3_digit	91.61	8.39	0.00	2,385
	4_digit	91.25	8.75	0.00	2,345
	5_digit	88.30	11.70	0.00	2,316
	6_digit	89.07	10.93	0.00	2,169
	10_digit	86.94	13.06	0.00	1,217

Table 6.25 Performance of Touching Digit Pairs (TDPs) extracted from NIST SD19

Reference	Recognition rate	Error rate	Rejection rate	# of Samples on the testing set
[CHI et al., 1995]	89.20	10.80	2.80	3,355
[HU & YAN, 1998]	89.66	10.34	NA	3,355
[ZHOU et al., 2000]	85.70	3.50	NA	4,395
Proposed method	89.61	10.39	0.00	2,069

7. Conclusions and Future Work

In this work, we have described a two-stage HMM-based method for recognizing handwritten numeral strings. With the first stage, we showed that the use of an implicit segmentation strategy is a promising way to deal with the string difficulties described in Chapter 1. The reason is that it avoids the need to define heuristics to group parts of broken digits or to separate touching digits, such as those used in the segmentation-based methods described in Chapter 2. However, there is some cost attached to this strategy related to the loss in terms of recognition performance caused by joining segmentation and recognition processes. During the experiments, it was possible to observe that the feature set and numeral models defined in the first stage (SCB), which have often been shown to be capable of finding the right segmentation points, are not strong enough in terms of recognition performance. The final SCB recognition rates of strings composed of 2, 3, 4, 5, 6, and 10 digits were 90.29%, 85.87%, 81.66%, 79.97%, 76.76% and 68.44% respectively. The average recognition rate of isolated digits was 94.00%. For touching digit pairs, this stage achieved a recognition rate of 79.51%.

With the second stage, we showed the contribution to handwritten numeral string recognition performance of considering a further verification step, which is used to re-rank the hypotheses of the first stage. Verification compensates for the loss in terms of recognition performance resulting from the necessary tradeoff between segmentation and recognition carried out in the implicit segmentation method.

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

The experiments considering the Verification Stage showed a significant improvement in the recognition performance. The average recognition rate of isolated

digits was 98.02%. The final recognition rate of known-length strings composed of 2, 3, 4, 5, 6, and 10 digits were 95.23%, 92.62%, 92.11%, 90.00%, 90.09% and 86.94% respectively. In addition, for touching digit pairs the method achieved a recognition rate of 89.61%. During these experiments, we observed that the use of a verification stage brought about an average improvement on the global string recognition rate of 9.92%.

The strategies used to consider unknown-length numeral strings showed a small loss in terms of recognition performance. The most promising results were obtained using the string length predictor. The final recognition rates for strings composed of 2, 3, 4, 5, 6 and 10 digits were 94.81%, 91.61%, 91.25%, 88.30%, 89.07% and 86.94% respectively. This means a loss in terms of global recognition rate of 0.91%.

We may improve the performance of the proposed method by further development in a number of areas. A simple improvement would be to develop a rejection mechanism. In contrast, we can improve the performance of the proposed method by further investigating feature sets, since this method enables the combining of different features at each stage. For instance, a new set of foreground features can be defined to improve the segmentation-recognition performance of the first stage, while new features with powerful recognition performance can be evaluated in the second stage.

In a similar way, further work can be carried out to improve the numeral models of each stage. For instance, in the first stage, it would be interesting to investigate a way of integrating additional contextual information into the numeral models regarding the interaction between adjacent numerals in strings. For this purpose, it could be interesting to build a pause instead of a space model into the numeral models. The same two-step training mechanism could be used to train this pause model. The idea is to pay some attention to the possibility of integrating handwriting-specific knowledge into the model structure to obtain a more accurate representation of numeral strings. We believe that, as the knowledge gained from ligatures and spaces between adjacent characters has been shown to be very important in increasing the word recognition performance in [CHO et al., 1995] and [DOLFING, 1998], the knowledge about overlapping, touching and spaces between adjacent numerals may play the same role for numeral strings.

These investigations of features and models can also take into account the development of character models instead of digit models. These can be used to extend the method to the recognition of handwritten words.

References

[BOSE & KUO, 1994] BOSE C.B. and KUO S-S. Connected and Degraded Text Recognition using Hidden Markov Model. *Pattern Recognition*, Vol. 27, No. 10, pp. 1345-1363, 1994.

[BOZINOVIC & SRIHARI, 1989] BOZINOVIC R.M. and SRIHARI S. Off-line Cursive Script Word Recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 1, pp. 68-83, Jan. 1989.

[BRITTO et al., 1999] BRITTO A.S., SABOURIN R., LETHELIER E., BORTOLOZZI F. and SUEN C.Y., Slant Normalization of Handwritten Numeral Strings, *Proceedings of the International Seminar on Knowledge Management/Document Management* (*ISKM/DM'1999*), vol 1, pp. 1-10, Curitiba-Pr, Brazil, December 1999.

[BRITTO et al., 2000] BRITTO A.S., SABOURIN R., LETHELIER E., BORTOLOZZI F. and SUEN C.Y. Improvement in Handwritten Numeral String Recognition by Slant Normalization and Contextual Information. *Proceedings of the Seventh International Workshop on Frontiers on Handwriting Recognition (IWFHR'2000)*, September 11-13, Amsterdam, The Netherlands, Vol. 1, pp. 323-332, 2000.

[BRITTO et al., 2001a] BRITTO A.S., SABOURIN R., BORTOLOZZI F. and SUEN C.Y., An Enhanced HMM Topology in an LBA Framework for the Recognition of Handwritten Numeral Strings, *Proceedings of the International Conference on Advances in Pattern Recognition (ICAPR'2001)*, Vol 1, pp. 105-114, Rio de Janeiro-Brazil, March 2001.

[BRITTO et al., 2001b] BRITTO A.S., SABOURIN R., BORTOLOZZI F. and SUEN C.Y., A two-stage HMM-based systems for recognizing handwritten numeral strings. To appear in *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR'01)*, Seattle, USA, September 2001.

[BROWN & GANAPATHY, 1983] BROWN, M.K. and GANAPATHY. Preprocessing Techniques for Cursive Word Recognition. *Pattern Recognition*, Vol 16, No. 5, pp. 447-458, 1983.

[CAI & LIU, 1998] CAI J. and LIU Z. Integration of Structural and Statistical Information for Unconstrained Handwritten Numeral Recognition. *Proceedings of the Fourteenth International Conference on Pattern Recognition (ICPR'98)*, Vol. I, pp. 378-380, 1998.

[CAO et al., 1995] CAO J., AHMADI M. and SHRIDHAR M. Recognition of Handwritten Numerals with Multiple Feature and Multistage Classifier. *Pattern Recognition*, Vol. 28, No. 2, pp. 153-160, 1995.

[CASEY & LECOLINET, 1996] CASEY R.G. and LICOLINET E. A Survey of Methods and Strategies in Character Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 7, pp. 690-706, July 1996.

[CHEUNG & YEUNG, 1998] CHEUNG K. and YEUNG D. A Bayesian Framework for Deformable Pattern Recognition with Application to Handwritten Character Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 12, December 1998.

[CHI et al., 1995] CHI Z., SUTERS M. and YAN. H. Separation of Single and Double Touching Handwritten Numeral Strings. *Optical Eng*., vol. 34, pp. 1159-1165, 1995.

[CHIM et al., 1998] CHIM Y.C., KASSIM A.A. and IBRAHIM Y. Dual Classifier System for Handprinted Alphanumeric Character Recognition. *Pattern Analysis & Applications*, 1, pp. 155-162, 1998.

[CHO et al., 1995] CHO W., LEE S.W. and KIM, J.H. Modeling and Recognition of Cursive Words with Hidden Markov Models, *Pattern Recognition*, 28:1941-1953, 1995.

[CORREIA & OLIVEIRA, 2000] CORREIA, S.E.N. and CARVALHO J.M. Optimizing the Recognition Rates of Unconstrained Handwritten Numerals using Biorthogonal Spline Wavelets. *Proceedings of the International Conference on Pattern Recongition*, Barcelona – Espanha, September, 2000.

[DELEVSKI & STANKOVIC, 1998] DELEVSKI V. and STANKOVIC S. Recognition on Handwritten Digits based on their Topological and Morphological Properties. *Advances in Pattern Recognition, Joint IAPR International Workshop SSPR'98 and SPR'98*, pp. 516-523, 1998.

[DOLFING, 1998] DOLFING J.G.A. *Handwriting recognition and verification: A hidden Markov approach*. PhD. thesis, Eindhoven University of Technology, The Netherlands, 1998.

[ELMS, 1996] ELMS A.J. *The representation and recognition of text using hidden Markov models* (PhD thesis). Department of Electronic and Electrical Engineering, University of Surrey, April 1996.

[ELMS et al., 1998] ELMS A.J., PROCTER S. and ILLINGWORTH J. The Advantage of using an HMM-based Approach for Faxed Word Recognition. *International Journal on Document Analysis and Recognition (IJDAR)*. pp. 18-36, 1998.

[FAVATA et al., 1994] FAVATA J.T., SRIKANTAN G. and SRIHARI S.N. Handprinted Character/Digit Recognition using a Multiple Feature/Resolution Philosophy. *Proceedings of the Fourth International Workshop on Frontiers in Handwriting Recognition (IWFHR-IV)*, pp. 57-66, 1994.

[FUJISAWA & NAKANO, 1992] FUJISAWA H. and NAKANO Y. Segmentation methods for character recognition: from segmentation to document structure analysis. *Proceedings of the IEEE*, vol. 80, pp. 1079-1092, July 1992.

[GADER, 1996] GADER P.D. Automatic Feature Generation for Handwritten Digit Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 12, pp. 1256-1261, December 1996.

[GADER & KHABOU, 1996] GADER P.D. and KHABOU M. A. Automatic Feature Generation for Handwritten Digit Recognition. *IEEE Transactions on Pattern Analysis and Machine Intellingence*, Vol. 18, No. 12, pp. 1256-1261, December 1996.

[GADER et al., 1997] GADER P.D., KELLER J.M., KRISHNAPURAM R., CHIANG J. and MOHAMED M.A. Neural and Fuzzy Methods in Handwriting Recognition. *Computer Magazine*, pp. 79-85, February, 1997.

[GILLOUX, 1994] GILLOUX M. Handwritten Digit Recognition Using Markov Meshes. *Proceedings of the Fourth International Workshop on Frontiers in Handwriting Recognition (IWFHR94)*, pp. 107-114, 1994.

[GUILLEVIC & SUEN, 1997] GUILLEVIC D. and SUEN C.Y. HMM Word Recognition Engine. *Proceedings of the Fourth International Conference on Document Analysis and Recognition (ICDAR97)*. pp. 544-547 Aug. 1997.

[GROTHER, 1995] GROTHER, P.J. *NIST Special Database 19 - Handprinted Forms and Characters Database*. National Institute of Standards and Technology (NIST), March 1995.

[HA et al., 1998] HA T.M., ZIMMERMANN M. and BUNKE H. Off-Line Handwritten Numeral String Recognition by Combining Segmentation-Based and Segmentation-Free Methods. *Pattern Recognition*, Vol. 31, No. 3, pp. 257-272, 1998.

[HEUTE et al., 1998] HEUTE L., PAQUET T., MOREAU J.V., LECOURTIER, Y. and OLIVIER, C. A structural/statistical feature-based vector for handwritten character recognition. *Pattern Recognition Letters*, 19 (1998), pp. 629-641.

[HIRANO et al., 1997] HIRANO T., OKADA Y. and YODA, F. Structural Character Recognition Using Simulated Annealing. *Proceedings of the Fourth International Conference on Document Analysis and Recognition (ICDAR'97)*, Vol. 2, pp. 507-510, 1997.

[HUANG, et al. 1990] HUANG X.D., ARIKI Y. and JACK M.A. *Hidden Markov Models For Speech Recognition*. Edinburgh Information Technology Series, Edinburgh University Press, Edinburgh, 276 p, 1993.

[HUANG et al., 1993] HUANG X.D., HON H.W., HWANG M.Y. and LEE K.F. A comparative study of discrete, semicontinuous, and continuous hidden Markov models. *Computer Speech and Language*, Vol 7, pp. 359-368, 1993.

[KEELER & RUMELHART, 1992] KEELER J. and RUMELHART D.E. A self-organizing integrated segmentation and recognition neural network. *Advances in Neural Information Processing Systems*, Vol 4., Moody J.E., Hanson S.J. and Lippmann R.P. eds., pp. 496-503, Morhan Kaufmann, San Mateo, CA, 1992.

[KIM et al., 1997] KIM J., SEO K. and CHUNG, K. A Systematic Approach to Classifer Selection on Combining Multiple Classifiers for Handwritten Digit Recognition. *Proceedings of the Fourth International Conference on Document Analysis and Recognition (ICDAR'97)*, Vol. 2, pp. 459-462, 1997.

[KIMURA & SHRIDHAR, 1992] KIMURA F. and SHRIDHAR M. Segmentation-recognition algorithm for handwritten numeral strings. *Machine Vision Applications*, No. 5, pp. 199-210, 1992.

[KIMURA et al., 1993] KIMURA F., SHRIDHAR M. and CHEN Z. Improvements of a Lexicon Directed Algorithm for Recognition of Unconstrained Handwritten Words. *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR'93)*, pp. 18-22, 1993.

[KIMURA et al., 1998] KIMURA F., INOUE S., WAKABAYASHI T., TSURUOKA S. and MIYAKE Y. Handwritten Numeral Recognition using Autoassociative Neural Networks. *Proceedings of the Fourteenth International Conference on Pattern Recognition (ICPR'98)*, pp. 166-171, 1998.

[LAM & SUEN, 1995] LAM L. and SUEN C.Y. Optimal combinations of pattern classifiers. *Pattern Recognition Letters*, Vol. 16, No. 9. pp. 945-954, 1995.

[LEE & GOMES, 1997] LEE L.L and GOMES N.R. Disconnected Handwritten Numeral Image Recognition. *Proceedings of the Fourth International Conference on Document Analysis and Recognition (ICDAR'97)*, Vol. 2, pp. 467-470, 1997.

[LEE & KIM, 1999] LEE, S-W. and KIM, S-Y. Integrated Segmentation and Recognition of Handwritten Numerals with Cascade Neural Network. *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, Vol. 29, No. 2, pp. 285-290, February 1999.

[LEE & PARK, 1994] LEE, S-W. and PARK J-S. Nonlinear Shape Normalization Methods for the Recognition of Large-Set Handwritten Characters. *Pattern Recognition*, Vol. 27, No. 7, pp. 895-902, 1994.

[LETHELIER et al., 1995] LETHELIER E., LEROUX M., and GILLOUX M. An Automatic Reading System for Handwritten Numeral Amounts on French Checks. *Proceedings of the Third International Conference on Document Analysis and Recognition (ICDAR'95)*, pp.92-97, 1995.

[LIM & CHIEN, 1998] LIM K. and CHIEN S. Neural Network-Based Feature Space Generation for Multiple Databases of Handwritten Numerals. *Proceedings of the Fourteenth International Conference on Pattern Recognition (ICPR'98)*, pp. 375-377, 1998.

[LIN et al., 1997] LIN X., DING X. and WU Y. Handwritten Numeral Recognition using MFNN-based Multiexpert Combination Strategy. *Proceedings of the Fourth International Conference on Document Analysis and Recognition (ICDAR'97)*, Vol. 2, pp. 471-474, 1997.

[LINDE et al., 1980] LINDE Y., BUZO A. and GRAY R.M. An algorithm for vector Quantization Design. *IEEE Transations on Communications*, vol. COM-28, pp. 84-95, Jan. 1980.

[LONCARIC, 1998] LONCARIC S. A Survey of Shape Analysis Techniques. *Pattern Recognition*, Vol. 31, No. 8, pp. 983-1001, 1998.

[MAKHOUL et al., 1995] MAKHOUL J., ROUCOS S., and GISH H. Vector Quantization in Speech Coding. *Proceedings of the IEEE*, vol. 73, pp. 1551-1558, Nov. 1985.

[MAKHOUL et al., 1998] MAKHOUL J., SCHWARTZ R., LAPRE C. and BAZZI I. A script-independent methodology for optical character recognition. *Pattern Recognition*, Vol 31. No. 9, pp. 1285-1294, 1998.

[MATAN et al., 1992] MATAN O., BURGES, Y., CUM L. and DENKER J.S. Multi-digit recognition using a space displacement neural network. *Advances in Neural Information Processing Systems*, Vol 4., Moody J.E., Hanson S.J. and Lippmann R.P. eds., pp. 488-495, Morhan Kaufmann, San Mateo, CA, 1992.

[MATIN et al., 1993] MATIN, G.L., RASHID M. and PITTMAN J.A. Integrated segmenation and recognition through exhaustive scans or learned saccadic jumps. *International Journal of Pattern Recognition and Artificial Intelligence*. Vol. 7, No. 4, pp. 831-847, 1993.

[NISHIDA & MORI, 1994] NISHIDA H. and MORI S. A Model-Based Split-and-Merge Method for Character String Recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, Vol 8, No. 5, pp. 1205-1222, 1994.

[NISHIWAKI & YAMADA, 1998] NISHIWAKI D. and YAMADA K. A New Numeral String Recognition Method using Character Touching Type Verification. *Proceedings of the Sixth International Workshop on Frontiers in Handwriting Recognition (IWFHR98)*, pp. 567-576, 1998.

[OLIVEIRA et al., 2000] OLIVEIRA, L.S, LETHELIER, E., BORTOLOZZI F., and SABOURIN R. A New Approach to Segment Handwritten Digits. *Proceedings of the Seventh International Workshop on Frontiers on Handwriting Recognition (IWFHR' 2000)*, September 11-13, Amsterdam, The Netherlands, Vol. 1, pp. 577-582, 2000.

[PARK & LEE, 1998] PARK H. and LEE S. A Truly 2-D Hidden Markov Model For Off-Line Handwritten Character Recognition. *Pattern Recognition*, Vol. 31, No. 12, pp. 1849-1864, 1998.

[PAVLIDIS, 1986] PAVLIDIS T. A vectorizer and feature extractor for document recognition. *Comput. Vision and Graphic Image Process*, 35, 111-127, 1986.

[PORITZ, 1988] PORITZ A.B. Hidden Markov Models: A Guided Tour. *Proceedings of the IEEE International Conf. On Acoustic, Speech, and Signal Processing (ICASSP88)*, pp.7-13, 1988.

[PROCTER & ELMS, 1998] PROCTER S. and ELMS A. J. The recognition of handwritten digit strings of unknown length using hidden Markov models. *Proceedings of the Fourteenth International Conference on Pattern Recognition (ICPR'98)*, pp. 1515-1517, 1998.

[RABINER, 1989] RABINER, L. R. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, Vol. 77, No. 2, pp.257-286, Feb. 1989.

[RABINER & JUANG, 1993] RABINER L.R. and JUANG B-H. *Fundamentals of speech recognition*. Prentice Hall Inc., Englewood Clifss, New Jersey, 1993.

[SABOURIN, 1990] SABOURIN, R. *Une Approache de Type Compréhension de Scene Appliqueé au Probleme de la Vérification Automatique de L'Identité par L'Image de la Signature Manuscrite*. These de Doctorate, Departement de Génie Électrique, École Politechnique, Université de Montreal.

[SHI et al., 1997] SHI Z., SRIHARI N., SHIN Y-C. and RAMANAPRASAD V. A System for Segmentation and Recognition of Totally Unconstrained Handwritten Numeral Strings. *Proceedings of the Fourth International Conference on Document Analysis and Recognition (ICDAR'97)*, Vol. 2, pp. 455-458, 1997.

[SHRIDHAR & BADRELDIN, 1986] SHRIDHAR M. and BADRELDIN A. Recognition of isolated and simply connected handwritten numerals. *Pattern Recognition*, vol. 19, pp. 1-12, 1986.

[SIMONCINI & KOVACS, 1995] SIMONCINI L and KOVACS V. A system for Reading USA Census 90 Handwritten Fields. *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR'95)*. Vol. 1, pp. 86-91, 1995.

[SUEN et al., 1992] SUEN, C.Y., NADAL C., LEGAULT, R., MAI T.A. and LAM L. Computer recognition of unconstrained handwritten numerals, *Proc. IEEE*, Vol. 80. No. 7, pp. 1162-1180, 1992.

[SUEN et al., 1998] SUEN, C.Y., LIU K. and STRATHY N.W. Sorting and Recognizing Cheques and Financial Documents. *Proceedings of Third IAPR Workshop on Document Analysis Systems (DAS'98),* pp. 1-18, 1998.

[TEO & SHINGHAL, 1997] TEO R.Y. and SHINGHAL R. A Hybrid Classifier for Recognizing Handwritten Numerals. *Proceedings of the Fourth International Conference on Document Analysis and Recognition (ICDAR'97),* Vol. 2, pp. 283-287, 1997.

[TRIER et al., 1996] TRIER O.D., JAIN A.K. and TAXT T. Feature Extraction Methods for Character recognition - a Survey. *Pattern Recognition*, Vol 29, No. 4, pp. 641-662, 1996.

[WANG, 1994] WANG X. *Durationally constrained training of HMM without explicit state duration PDF*. Institute of Phonetic Sciences, University of Amsterdam, Proceedings 18, pp. 111-130, 1994.

[WANG et al., 1998] WANG X., GOVINDARAJU V. and SRIHARI S. Holistic Recognition of Touching Digits. *Proceedings of the Sixth International Workshop on Frontiers in Handwriting Recognition (IWFHR98),* pp. 295-303, 1998.

[WESTALL & NARASIMHA, 1993] WESTALL J.M. and NARASIMHA M.S. Vertex Directed Segmentation of Handwritten Numerals. *Pattern Recognition,* Vol. 26, No. 10, pp. 1473-1486, 1993.

[XU et al., 1992] XU L., KRZYZAK A. and SUEN C.Y. Method of combining multiple classifiers and their application to handwritten numeral recognition. *IEEE Trans. on Systems, Man and Cybernetics*, vol. 22, pp. 418-435, 1992.

[YABOUBI, 1996] YACOUBI A. E. *Modélisation Markovienne de l'écriture manuscrite application à la reconnaissance des adresses postales*. (These doctorate). Universite de Rennes 1, 1996.

[YOON et al., 2000] YOON S., KIM G., GHOI Y. and LEE Y. Handwritten numeral string recognition using continuation property. *Proceedings of the Fourth International Workshop on Document Analysis Systems (DAS' 2000).* December 10-13, Rio de Janeiro, Brazil, pp. 167-176, 2000.

[YU & YAN, 1998] YU D. and YAN H. Separation of Single-Touching Handwritten Numeral Strings Based on Structural Features. *Pattern Recognition*, Vol. 31, No. 12, pp. 1835-1847, 1998.

[ZHANG et al., 1998] ZHANG B., FU M. and YAN H. A Modular Classification Scheme with Elastic Net Models for Handwritten Digit Recognition. *Proceedings of the Fourteenth International Conference on Pattern Recognition (ICPR'98)*, pp. 1859-1861, 1998.

[ZHOU et al., 1997] ZHOU J., QIANG G. and SUEN C.Y. A High Performance Hand-printed Numeral Recognition System with verification Module. *Proceedings of the Fourth International Conference on Document Analysis and Recognition (ICDAR'97),* Vol. 2, pp. 293-297, 1997.

[ZHOU & SUEN, 2000] ZHOU J. and SUEN C.Y. Recognition and Verification of Touching Handwritten Numerals. *Proceedings of the Seventh International Workshop on Frontiers on Handwriting Recognition (IWFHR'00),* September 11-13, Amsterdam, The Netherlands, Vol. 1, pp. 179-188, 2000.

A. Appendix – Extraction of numeral strings from the NIST SD19 database

In this appendix, the processes used for extracting numeral strings from the full-page forms available in Special Database 19 (SD19) of the National Institute of Standard Technology (NIST) database are described. With the handwritten numeral strings extracted from these forms, we created a database called NString_SD19, which is valuable for training, validating and testing recognizers that go beyond isolated digit classification.

A detailed description of the origin, publication history and organization of NIST SD19 is available in [Grother 95]. In Section A.1, we present a brief description of this public database. In Section A.2, the process used to extract the numeral strings from the NIST database is described. In Section A.3, we present the process used to provide an isolated digit database in which each digit sample has a link with its original string. Finally, in Section A.4, the structure and content of the new NStringSD19 database are presented.

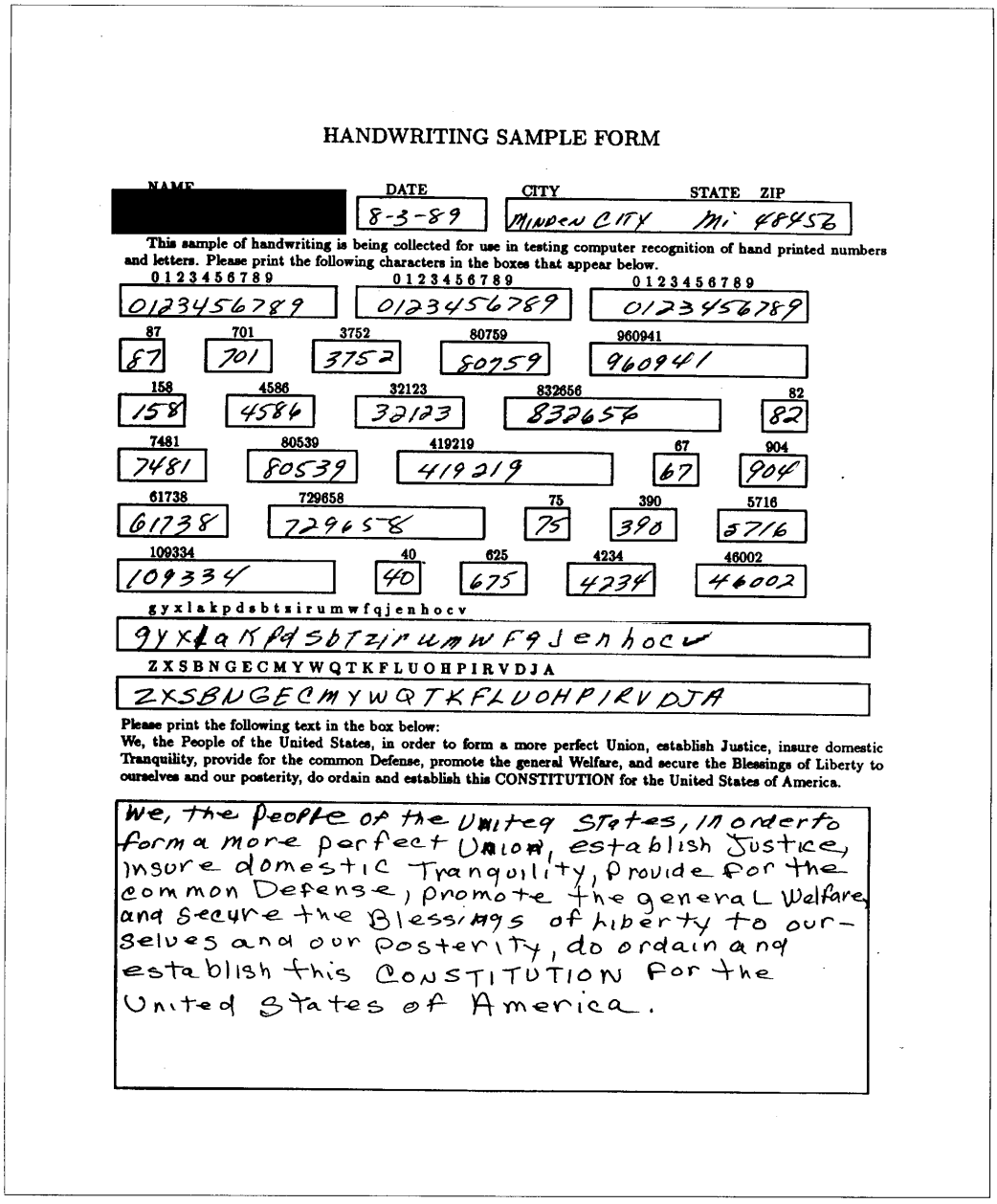
A.1. NIST Special Database 19 (SD19)

The SD19 is composed of 3669 full-page binary images of Handwritten Sample Forms (HSF), which are organized in eight series, denoted by hsf_{0,1,2,3,4,6,7,8}. A total of 814,255 handwritten labeled characters (digit and alphabetic) have been segmented from these forms and organized by class, field and writer (upper and lower cases are merged). These isolated characters, as well as the full-page images, can be found on the original SD19 compact disc.

Table A-1. HSF series distribution

Series	Number of HSF pages
hsf_0	500
hsf_1	500
hsf_2	500
hsf_3	600
hsf_4	500
hsf_6	499
hsf_7	500
hsf_8	70
Total	3669

Table A.1 shows the distribution of HSF pages per series. Usually, the hsf_{0,1,2,3,6,8} are used for training purposes, hsf_7 for validation and hsf_4 for testing. Note that hsf_5 does not exists in SD19.


HANDWRITING SAMPLE FORM

NAME [redacted] DATE 8-3-89 CITY MINDEN CITY STATE MI ZIP 48456

This sample of handwriting is being collected for use in testing computer recognition of hand printed numbers and letters. Please print the following characters in the boxes that appear below.

0 1 2 3 4 5 6 7 8 9 0123456789 0123456789

87 701 3752 80759 960941

158 4586 32123 832656 82

7481 80539 419219 67 904

61738 729658 75 390 5716

109334 40 625 4234 46002

g y x l a k p d s b t z i r u m w f q j e n h o c v

ZXSBNGECMYWQTKFLUOHPIRVDJA

Please print the following text in the box below:
We, the People of the United States, in order to form a more perfect Union, establish Justice, insure domestic Tranquility, provide for the common Defense, promote the general Welfare, and secure the Blessings of Liberty to ourselves and our posterity, do ordain and establish this CONSTITUTION for the United States of America.

We, the People of the United States, in order to form a more perfect Union, establish Justice, insure domestic Tranquility, provide for the common Defense, promote the general Welfare, and secure the Blessings of Liberty to ourselves and our posterity, do ordain and establish this CONSTITUTION for the United States of America.

Figure A-1 Handwriting Sample Form (HSF full-page form)

An example of a full-page NIST form or HSF page is shown in Figure A-1. We can see that an HSF page consists of 34 fields, 28 of which contain only numeric characters. The field descriptions are presented in Table A-2.

Table A-2 Handwriting sample form (HSF) fields

Field	Description
fld_0	Name
fld_1	Date
fld_2	City/State/ZIP
fld_3 …fld_30	Numeric field
fld_31	Lower case character box
fld_32	Upper case character box
fld_33	Free format text

A total of 100 HSF templates were used to fill up the HSF pages. The number, size and location of the fields are the same in all template variations. However, they present different strings of characters. These templates are provided by NIST SD19 in the form of truth files "refxx.txt", where xx represents a NIST template from 00 to 99. Similarly, the page image files (or forms) have name of the form "fyyyy_xx.tif", where yyyy identifies the writer and xx the template number.

Numeric Fields

Each HSF page is composed of 28 numeric fields (fields fld_3 to fld_30), filled out with numeral strings of different lengths. There are strings of 2, 3, 4, 5, 6 or 10[1] digits. Table A-3 presents the HSF numeric fields, their respective string length and the total number of strings extracted from the HSF pages using the process described in this Appendix.

Table A-3 Field number, respective string length and number of samples

Field Number	String length	Total number of strings
fld_3, fld_4, fld_5	10	11,007
fld_6, fld_15, fld_19, fld_23, fld_27	2	18,345
fld_7, fld_11, fld_20, fld_24, fld_28	3	18,345
fld_8, fld_12, fld_16, fld_25, fld_29	4	18,345
fld_9, fld_13, fld_17, fld_21, fld_30	5	18,345
fld_10, fld_14, fld_18, fld_22, fld_26	6	18,345
Total		102,732

[1] The 10 digit string is always composed of the digits from 0 to 9 in that order and is invariant from one template to another.

The labels of these fields are found in the reference files (templates) provided by NIST, where each line consists of a field name and the data that the writer was instructed to print in it.

A.2. The Numeral String Database (NString_SD19)

The process used to extract the numeral strings from the HSF pages is based on two steps: 1) field extraction from the HSF page; 2) preprocessing for bounding box deskewing and removal. In the first step, we detect and extract the field boxes from a page and save them separately as new files of the form "fyyyy_xx_w.tif" where yyyy and xx identify, as before, the writer and the template, and w corresponds to the field number from 3 to 30. To achieve this, a process based on vertical and horizontal projections is used to locate the coordinates of the upper-left corner of the field *fld_0*. The coordinates of the other fields are calculated using the *fld_0* coordinates as a reference.

In order to extract the numeral strings from these cut fields (102,732), the preprocessing first deskews the field bounding box. Then, vertical and horizontal projections are used locate the lines of the field bounding box in order to remove them. The resulting images contained only the cleaned numeric strings files are saved as "cdfyyyy_xx_w.tif" (cd = cleaned and deskewed[2]).

A.2.1. Field extraction from the HSF page

This segmentation process consists of locating and extracting all the numeric fields from the HSF pages, using the position of the first HSF field (*fld_0*) as a reference. Considering that the number, size and location of the fields are the same among all the HSF template variations, a data structure composed of line and column offset values for each numeric field from *fld_0* is used. This data structure is initialized for a given HSF page sample using the distance of the upper left corner coordinates of each numeric field from the upper left corner coordinates of *fld_0*. This relative distance

[2] The deskewing was only applied on the field bounding box. No character deskewing or slant correction was performed.

reduces the effects of possible variability in the position of the HSF pages during the scanning process.

The first step of the process is to locate the field *fld_0*. Then, the upper left coordinates of that field and the offset values in the data structure are used to calculate the position of each numeric field. The search for the *fld_0* upper left corner is performed in a specific region defined using 30% of the HSF page height, and 50% of the HSF page width. The algorithm searches for a horizontal single black run with a size greater than a parameter HS (30% of the *fld_0* width). The line position of this horizontal single black run is used as the *x* coordinate of fld_0. Then, in order to find the *y* coordinate, a search for a vertical line is performed which is considered to start in *x* with a size greater than VS (60% of the *fld_0* height). Figure A-2 shows some field images extracted from an HSF page.

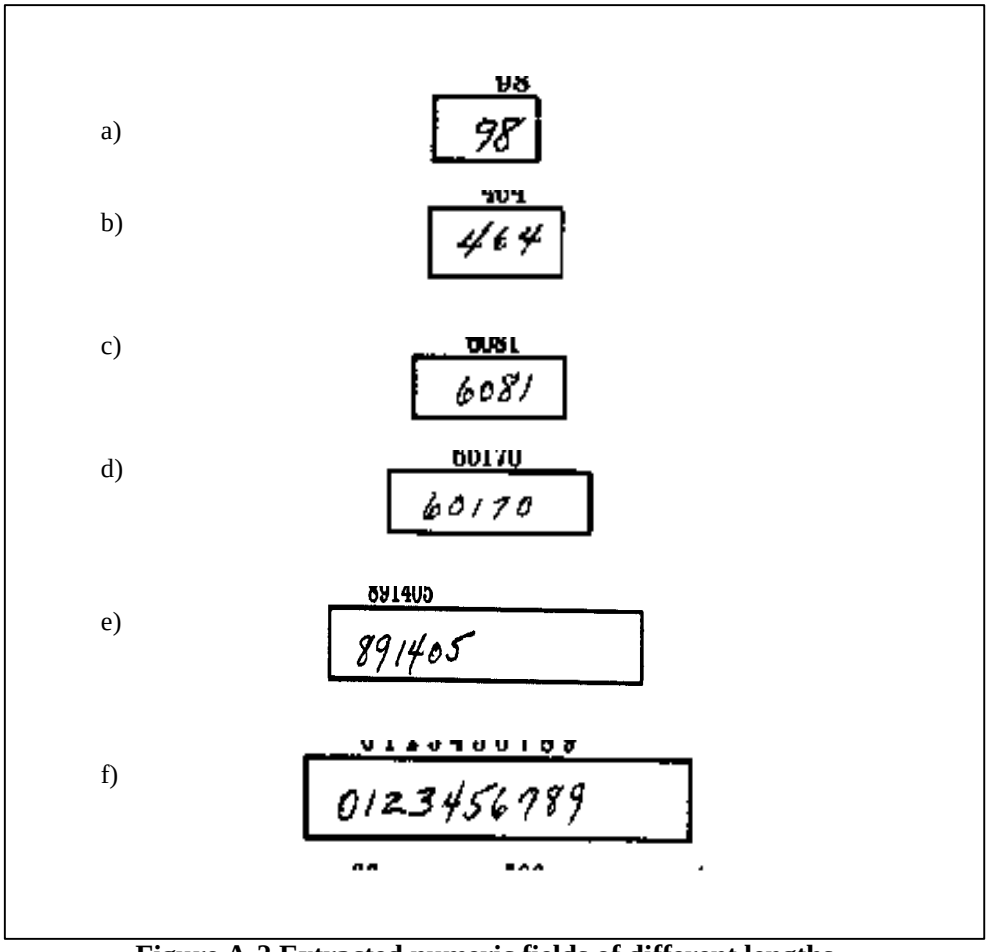


Figure A-2 Extracted numeric fields of different lengths

A.2.2. Preprocessing for bounding box deskewing and removal

After extracting the numeric fields from the HSF page, a field filtering process is used to eliminate the bounding box and the additional information outside it, which are considered as noise. The result of this process is an image containing just the handwritten numeral string. The algorithm is composed of two steps: skew correction and noise removal.

A.2.2.1. Skew correction

This preprocessing step is necessary since the process used to remove the field bounding box is based on vertical and horizontal projections. The algorithm used for skew correction can be summarized as follows:

1) Skew estimation

- Vertical projection (VP) is calculated;
- The peak corresponding to the bottom line of the bounding box is found in VP;
- A region around the peak is defined as the bottom line position. For this, all the entries adjacent to the peak with size greater than 40% of the peak size are considered to be part of the bottom line. The estimated bottom line position is used to calculate the bounding box skew (slope a);

2) Skew Correction

The slope a is used to rotate the field image. The (x, y) coordinates of each black pixel are replaced by the coordinates (x', y'), as follows:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos a & -\sin a \\ \sin a & \cos a \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} \tag{A.1}$$

A.2.2.2. Noise removal

This step eliminates the bounding box and all the information outside it. It is based on horizontal and vertical projections of the black pixels (histograms). The peaks

of these projections are used as limits to erase the bounding box and the surrounding noise. The filtering process first erases the horizontal lines, as follows:

- Horizontal projection (HP) is calculated;
- Two peaks (ps, pi) corresponding to the top and bottom horizontal lines of the field bounding box are located in HP;
- The internal limits of these lines (ls, li) are also found in HP. All entries adjacent to the peaks, greater than 40% of the peak size, are considered to be part of the horizontal lines (see Figure A-3);
- All information from the internal limit (ls) is erased. A process similar to this is applied from the internal limit (li).

Subsequently, another similar process is applied by which the vertical projection is used to erase the vertical lines of the bounding box. Some examples of numeric fields after preprocessing are shown in Figure A-4.

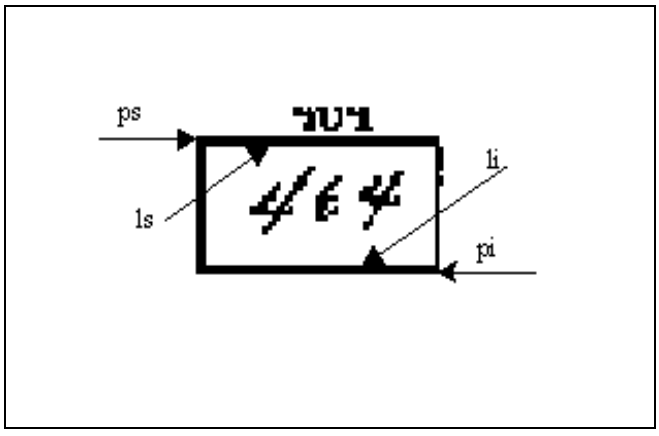


Figure A-3 Field and respective ps, pi, ls and li locations

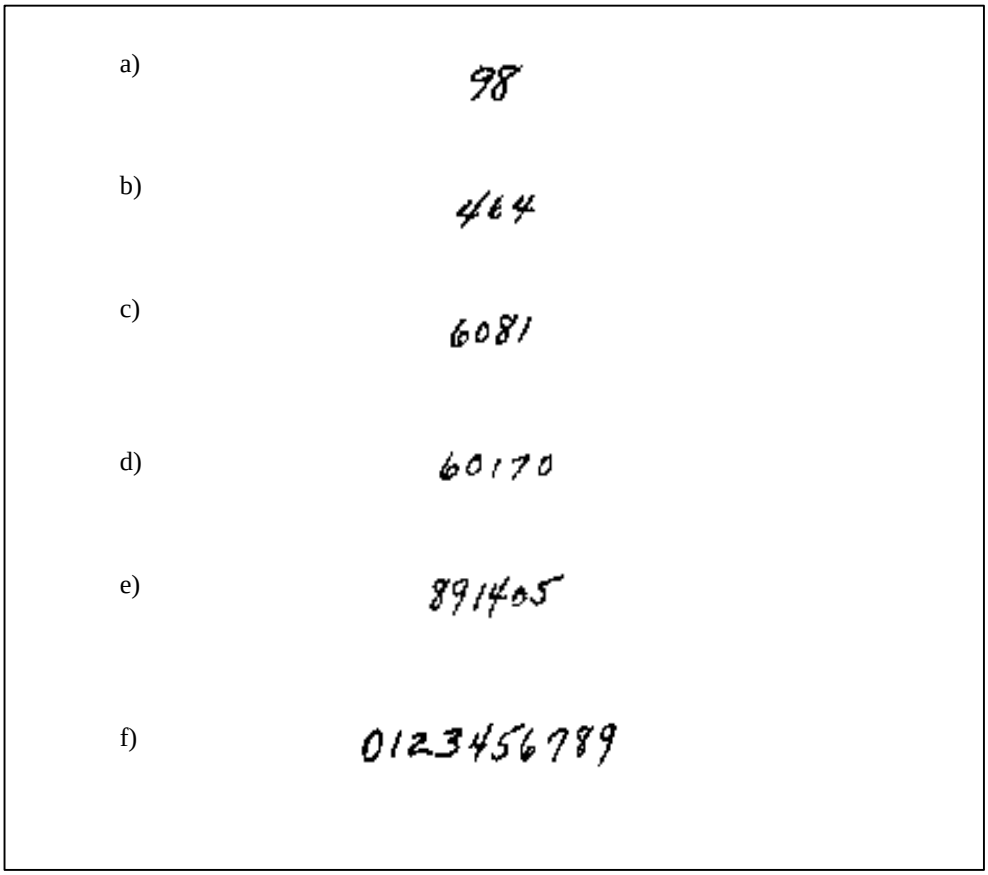


Figure A-4 Numeric field images after preprocessing

A.2.3. Semi-automatic checking process

The numeral strings were then automatically classified using a process based on isolated digit recognition. Each CC (connected component) in a given numeral string is submitted to the CENPARMI digit recognizer. If all the CCs are recognized as digits, and the global string recognition result is the same as that provided in the NIST reference file, the numeral string is considered to be a naturally segmented sample. These strings are used during the preliminary analyses of the NIST database, when some important observations are made, such as numeral height variation and the effect of slant correction on overlapping numerals.

The remaining strings were manually verified and classified according to the difficulties of recognition: touching, fragmented, noise and other. As a by-product of the automatic process for numeral string classification, we obtained an isolated digit database, in which there is a link between an isolated digit image and its original

numeral string. This link is provided by the image file names. The image file name refers to an isolated digit in the form: cdfyyyy_xx_w_z.tif", where we have the original numeral string image name plus 'z', where z corresponds to the digit position in the string. These numerals are also used in the preliminary analyses to allow some testing of slant-correction of isolated numerals, using the slant estimated from the original strings. Figure A-5 shows an overview of the process used to select naturally segmented numeral strings and isolated digits linked to them.

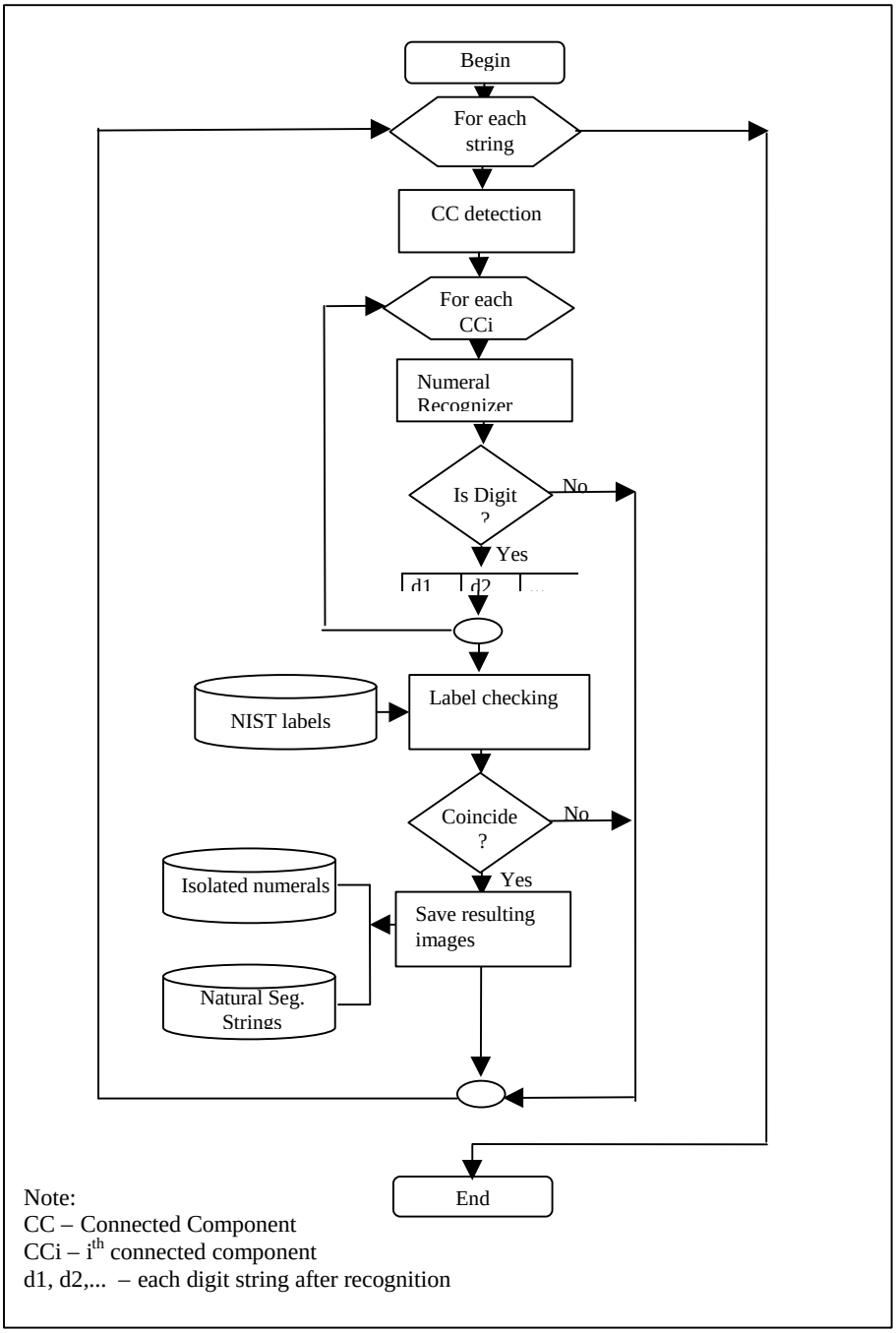
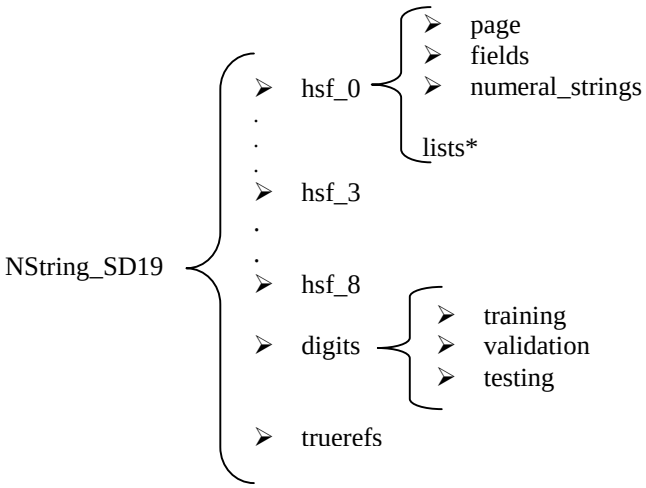


Figure A-5 Process to select naturally segmented strings and corresponding isolated digits

A.2.4. NString_SD19 organization and contents

The structure of NString_SD19 is shown in Figure A-6. We have in the hsf_series directory one subdirectory for each NIST series, a subdirectory for the reference files and a subdirectory for the isolated digits.



* list of file names saved as text files.

Figure A-6 Structure of NString_SD19

Each HSF series subdirectory is divided into 3 sections:

- hsf_x/pages: contains a copy of the original full-page images;
- hsf_x/fields: contains the original numeric fields extracted from each HSF page.
- hsf_x/numeral_strings: contains the numeral string extracted from the field images.

where x is the series number.

In the numeral_string subdirectory, we can find two groups of file name lists. A first group is composed of the following lists: 2_digit.list, 3_digit.list, 4_digit.list, 5_digit.list, 6_digit.list, 10_digit.list. These lists are used to classify the numeral strings by length. The second group is composed of lists used to classify the numeral strings by quality, as:

- natseg.list: contains naturally segmented numeral strings;
- touch.list: contains numeral strings with touching numerals;

- fragm.list: contains numeral strings with fragmented numerals. The fragmentation can be caused by the scanning process or by handwriting style;
- noise.list: contains numeral strings damaged by the field extraction process, or filled out twice by the writer;
- others.list: contains numeral strings in which one or more digits were not recognized by the classifier.

The *digits* directory is divided into 3 sections: training, validation and testing (which contains handwritten isolated digits extracted respectively from hsf_{0, 1, 2,3}, hsf_7 and hsf_4). Table A-4 shows the distribution by digit class in each of these sections, while Table A-5 summarizes the number of strings extracted from each series of NIST_SD19 and their classification by quality, as: naturally segmented, touching, fragmented, noise and other.

Table A-4 Distribution by digit class

Class	training hsf_{0, 1, 2, 3}	Validation hsf_7	Testing Hsf_4
0	19,776	4,629	2,899
1	20,966	5,010	3,231
2	19,608	4,675	2,958
3	19,766	4,692	3,060
4	19,652	4,624	2,869
5	17,532	4,187	2,726
6	19,993	4,715	3,044
7	20,509	4,933	3,009
8	19,866	4,681	2,978
9	20,116	4,860	2,979

Table A-5 Number of strings extracted from each NIST series and their classification by quality

Series	Number of numeral strings	Naturally segmented	Touching	Fragmented	Noise	Other
Hsf_0	14,000	10,353	1,081	1,027	840	699
Hsf_1	14,000	10,325	961	1,152	934	628
Hsf_2	14,000	10,007	801	1,342	889	961
Hsf_3	16,800	12,511	799	1,549	998	943
Hsf_4	14,000	6,485	1,282	1,946	878	3,409
Hsf_6	13,972	10,519	943	1,086	597	827
Hsf_7	14,000	10,194	1,052	884	465	1,405
Hsf_8	1,960	1,501	105	127	55	172
Total	102,739	71,895	7,024	9,113	5,656	9,044

A.2.5. Summary

In this appendix the process used to create a new database from NIST_SD19 is described. With the numeral strings extracted from the HSF pages we create the Nstring_SD19 database, which is usefull for evaluating handwritten numeral string recognition methods.

Notice that all processes used to extract, deskew, and clean the fields were strongly based on heuristics, since our main objective was just to obtain enough numeral string samples for constructing and evaluating the method proposed in this thesis. Even the explicit segmentation used to extract the string digits in order to submit them to the CENPARMI classifier is a simple connected component detection, since at that time, the proposed method was under_construction. The objective of this process was just to reduce efforts of the manual check, and also extracting some samples of naturally segmented strings and their corresponding digits to be used in the preliminary analyses presented in Chapter 4. We can say that a by_product of this thesis is the creation of the NStringSD19 database which is available in CD. The work on the NStringSD19 database was done in parallel to the construction and evaluation of the proposed method for recognising numeral strings. This means that we did not use all the benefits of this database, such as the previously knowledge about the number of touching, framented and noise images in each series of the NIST database.